VOSS Automate
Advanced Configuration Guide

Release 25.3

December 03, 2025

Legal Information

DOCUMENT ID: 20251203072443

Contents

1. What's New

1.1. Advanced Configuration Guide: Release 25.3

- EKB-24170: Improve Action Search list details. See: *Use the Action search to navigate Automate*

  Updated Action search topic with improvements, specifically, matching of queries, and ability to sort, filter, and clear results.

- EKB-26195: Macro query using wildcard (*) should return all fields as well as __pkid. See: *Dot notation*

  Updated macro syntax docs to describe the use of wildcards with _pkid in a macro query to return instances with their unique ID.

- EKB-26195: Macro query using wildcard (*) should return all fields as well as __pkid. See: *SELECT FROM WHERE macro syntax*

  Updated macro syntax docs to describe the use of wildcards with _pkid in a macro query to return instances with their unique ID.

- VOSS-1464: System User Audit and cleanup process. See: *Settings*

  Added details on the new user audit feature

- VOSS-1507: (EKB-24159: Create new Global settings for enabled services). See: *Global settings*

  Added details on the new Microsoft Defender for Office & Endpoint functionality.

- VOSS-1545: Configurable Microsoft Sync. See: *Global settings*

  Added details for the vendor-agnostic Datasync Builder tool, templates, and profiles, which are also integrated with the MS tenant.

- VOSS-1560: Microsoft Tenant App Registration Enhancements. See: *Global settings*

  Changes when adding and editing a Microsoft tenant.

- VOSS-1590: Microsoft: Enhance update sync FTP/Move via Data Sync and Manual Move (Move User). See: *Global settings*

  Docs updated for flow through provisioning update sync for MS user moves, line configuration changes on MS move user and services, and global settings (FTP).

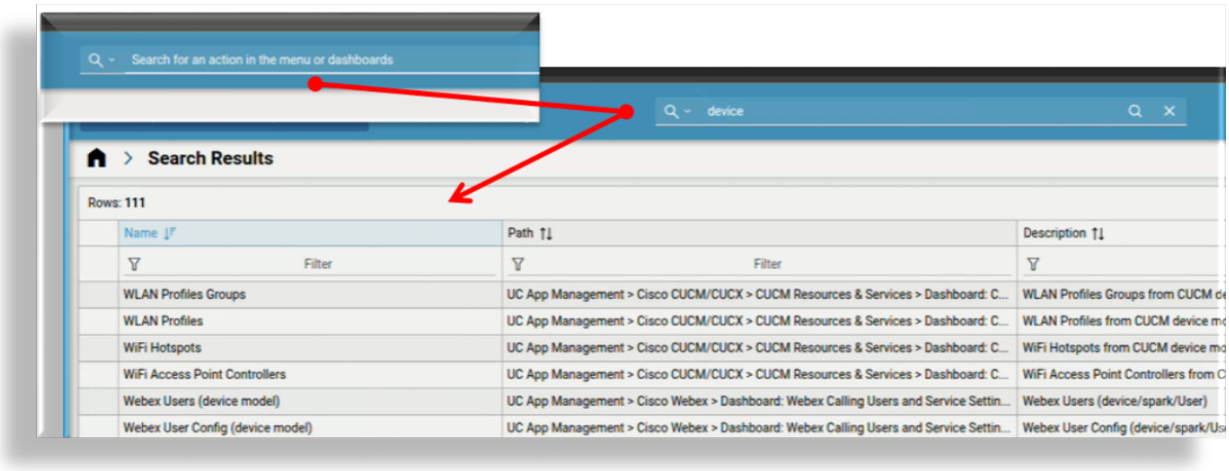
2. Settings and Tools

Important: User who have access to and modify `data/Settings` entries should back these up (for example, export the Global instance JSON) prior to upgrading and subsequently restore the backup (for example, import the exported Global instance).

2.1. Use the Action search to navigate Automate

2.1.1. Overview

Automate's toolbar **Action search** helps you to quickly find functionality, dashboards, and menu items in the Automate GUI. It's the default search method and is especially useful when searching for functionality in a role-based dashboard layout.



Related topics

- Search in Automate in the Core Feature Guide

2.1.2. How to use Action search

1. Type the name of the page, dashboard, or action you're looking for,
2. Press **Enter** or click the **Search** icon.
3. Select the relevant result to open the page.

2.1.3. Result handling

Action search uses title text from menus, dashboards, and actions, and also matches against model names, even if those names aren't visible in the result string. This improves relevance but may occasionally surface unexpected results. To ensure clarity, queries that match only the underlying model type (without a friendly label) are excluded from results.

The result set includes functionality that makes it easier to manage large result sets and to quickly locate a desired action:

- **Sortable and filterable column headers**: You can sort results alphabetically or reverse-alphabetically.
- **Keyword filters**: Enter filter criteria to narrow down long result lists.
- **Clear filters**: Reset filters to view the full result set again.

2.1.4. Result formats

- Menu entries appear as: *Menu > Submenu*
- Dashboard entries appear as: *Dashboard: Name*

2.1.5. Search tips

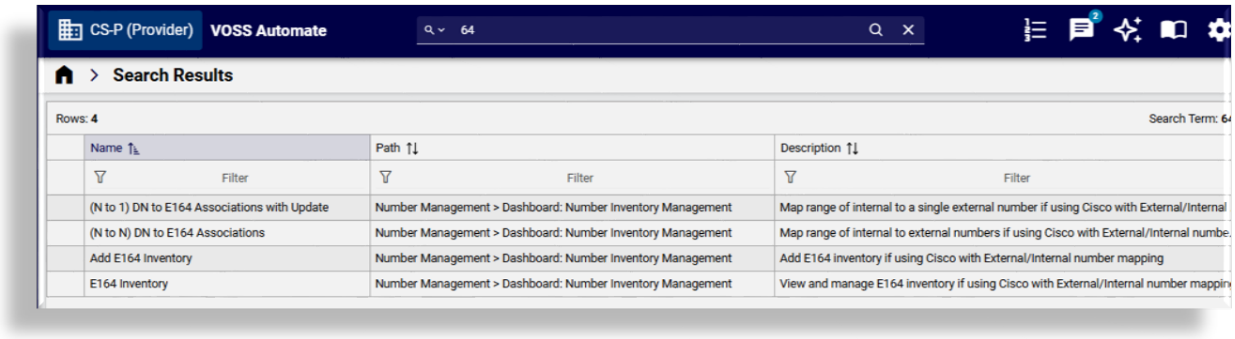
To improve your Action search results:

- Use singular keywords, for example, "phone" instead of "phones", or *add subscriber* instead of *add subscribers*.
- Avoid filler words like "the", "a", or "an".
- Use specific verbs like "create", "add", "update", "edit", "delete", or "remove".

  Note: Verbs in the search phrase return results that match the same word in a menu or dashboard, based on your role permissions.

- Include device names, such as "Cisco UCM", to find related models and actions.
- Abbreviations are supported. For example, "QAS" finds "QAS - MS Teams" and "Quick Add SIP Gateway".

- Include digits to find menus with numbers in their names, for example, “E164 Inventory”.



2.1.6. Example use cases

- Searching for “SIP Gateway” filters out unrelated models unless explicitly named.
- Typing “Cisco” and sorting results helps quickly locate all Cisco-related actions and dashboards.
- Filtering by “add” or “edit” helps isolate tasks based on operation type.

2.1.7. Best practices for admins

To ensure meaningful search results with Action search:

- Use familiar and descriptive terms when naming menus, dashboards, and actions.
- Configure dashboards based on user roles for faster access.
- Action search can be used to locate tasks not visible in dashboards or menus.
- Regularly review naming conventions to align with user expectations and improve discoverability.

2.2. Login notification banner

A banner, typically a security notice or user agreement, can be configured at a hierarchy level to show on the Administrator and Self-service login page before login.

High level administrators who have access to the `data/LoginBanner` model can configure the banner. A banner can be created so that:

- Only one instance is allowed per hierarchy

If an administrator or Self-service user logs in and belongs to a hierarchy for which there is no defined login banner, the first banner higher up on the hierarchy is displayed. If no banners are configured, then the user logs in without a banner.

The banner text is displayed in the format that it is entered into the input box upon configuration.

When the banner is configured, users will see the banner displayed on the login page after they enter their credentials and when they click the **Login** button. An **Agree** and **Cancel** button is shown beneath the banner. Users then need to click the **Agree** button to complete the login. If they click **Cancel**, they are returned to the login page.

Note: This banner is independent of the text on the login screen that may contain a privacy policy reference. The privacy policy text and reference on the login page is configured as a part of the Login Page Details when managing a theme.

2.3. Transaction log levels

Automate users with access to the `data/Settings` model or `view/DataSettings` can manage global transaction logging levels. These logging messages display below a selected transaction on the **Transactions** page, and have no impact on the transaction or sub-transaction action or detail information.

Log levels are *cumulative*, that is, more detailed levels include all details from less detailed levels.

The table describes the transaction log levels and their severity values:

Level	Description	Severity
Disabled	Disables all transaction log messages	999
Error	Only displays error messages	40
Warning	Also adds warning messages to above	30
Info	Also adds informational messages to above	20
Verbose	Also adds messages used for diagnostic purposes to above	15
Debug	Also adds advanced diagnostic messages - for future use	10

- Severity values are referenced (from value - to value) in transaction details when the log level is changed on this setting or changed by lower level administrators from the **System Settings** menu. See: *Settings*.
- If a transaction fails, the Log block includes all entries with severity values larger than that of the default or selected level of logging.
- The log levels of data syncs can be set to override these global levels.

  The transaction log level used for data sync and its immediate sub-transactions is by default set to Warning when it is not set.

  For details, refer to the Create a Custom Data Sync topic in the Core Feature Guide.

Transaction log levels are configurable based on your needs, for example, set to `Verbose` to include greater details for immediate troubleshooting and customization work:

1. Toggle the log level to `Verbose`.
2. Reproduce the issue causing the problem.
3. Export the logs and forward them to VOSS support.
4. Toggle the log level back to `Info`.

Note: When setting transaction log level to `Verbose` or `Debug`, notes and warnings are shown to remind admins that the retention period of such logs is due to their increased size, which consequently reduces the date range of available logs for troubleshooting.

2.4. Settings

sys-admin

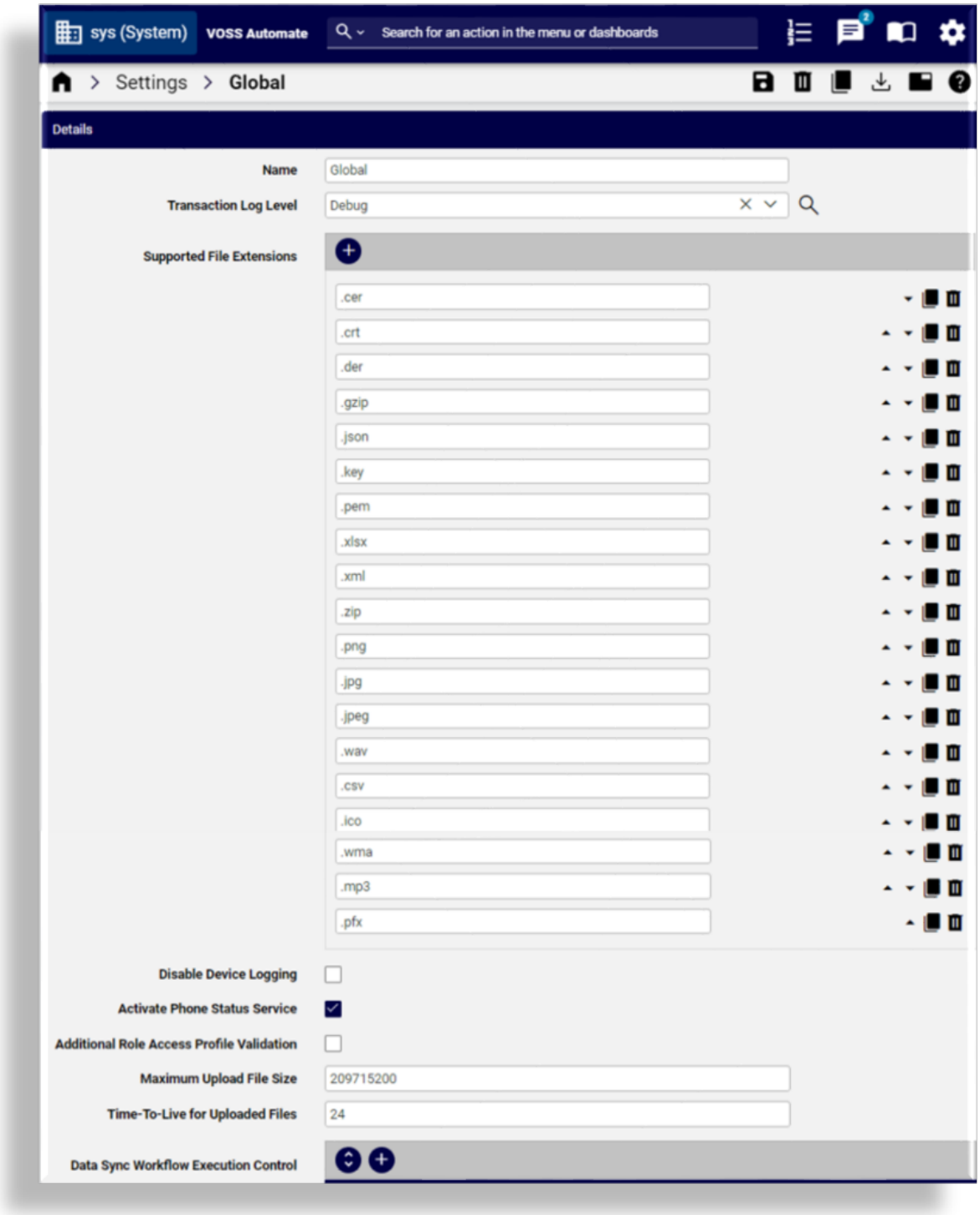
Tip: *Use the Action search to navigate Automate*

2.4.1. Overview

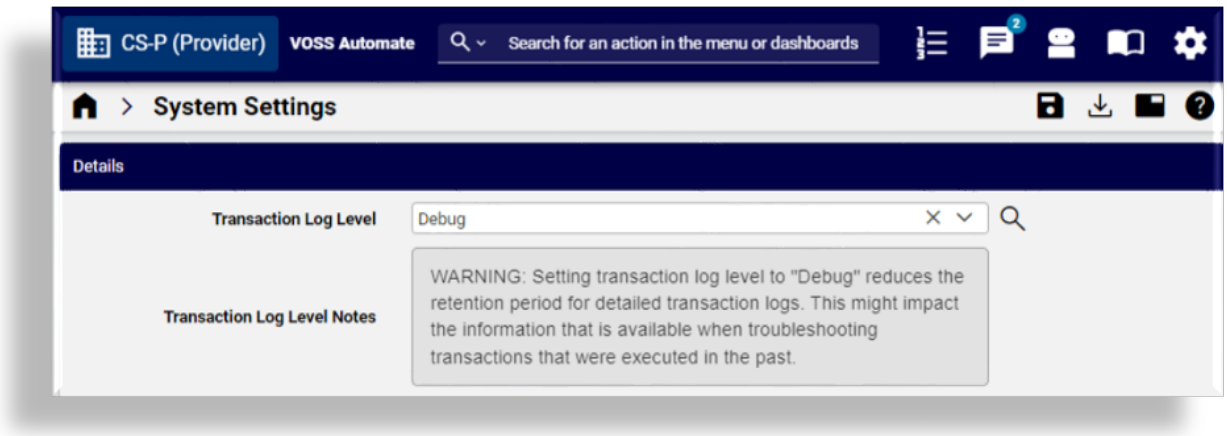
sys-admin

This topic describes the system level settings configurable via the **Settings** page, where a `sysadmin` user with access to the `data/Settings` model can view and modify the following global system level settings for Automate:

- Configure transaction log levels (See *Configure transaction log levels*)
- Add or remove supported file extensions (See *Supported file extensions*)
- Enable or disable system user audit (See System User Audit in the Core Feature Guide)
- Enable or disable device logging (See *Disable device logging*)
- Enable or disable the phone status active service (the service to periodically fetch Call Manager phone status). See *Activate phone status service*
- Enable or disable role access profile validation (See *Additional role access profile validation*)
- Specifying the maximum file upload size (See *File upload limitations*)
- Specify the time to live for uploaded files, in hours (See *Time-to-live for uploaded files*)
- Configure controls for data sync workflows on certain models via allowlist and denylist attributes (See *Data sync workflow execution control*)
- Configure device overwrite check exemptions on certain models via attributes (See *Device overwrite check exemptions*)
- Enable or disable Wingman at the system level (the Automate chat co-pilot)
- Choose when tooltips display (for accessibility)



Note: Provider admins and admins at `hcsadmin` have access to the **System Settings** page (`view/DataSettings`) for managing transaction log levels only, and is a limited version of `data/Settings`.



2.4.2. Configure transaction log levels

System transaction log levels refer to the configured verbosity level of the transaction logs in Automate. Available transaction log levels for Automate are as follows:

- Debug
- Verbose
- Info (default)
- Warning
- Error
- Disabled

Note: See *Transaction log levels* for details.

The following Automate admin users can view and modify transaction log levels in Automate:

- Provider admins and `hcsadmin` admins with access to the `view/DataSettings` model
- High level admins (`sysadmin`) with access to the `data/Settings` model

2.4.3. Supported file extensions

Users with system administrator privileges to the `data/Settings` model, typically, `sysadmin`, can manage valid file extensions in Automate.

By default, the following file extensions are specified as supported in the Global instance of `data/Settings`, which means that files with these extensions can be uploaded to Automate.

- `.cer`
- `.crt`
- `.der`
- `.gzip`

- .json
- .key
- .pem
- .xlsx
- .xml
- .zip
- .png
- .jpg
- .jpeg
- .wav
- .csv
- .ico

The sysadmin user can add or remove file extensions. Unsupported files can't be uploaded or imported, and will trigger a system error.

2.4.4. Disable device logging

Users with system administrator privileges to the data/Settings model, typically, sysadmin, can enable or disable device logging.

When setting **Disable Device Logging** to enabled (on the **Settings** page), then UCM AXL (and other external calls) requests and responses aren't written to the transaction log, and neither are generic driver requests, responses, template evaluations, and macro evaluations. These details won't display for the relevant transactions (on the **Transaction Log** page).

2.4.5. Activate phone status service

High level admins (sysadmin) with access to the data/Settings model (**Settings** page) can view and update the **Activate Phone Status Service** checkbox, which is enabled by default.

When enabled, this setting indicates that the real-time information (RIS) data collector service is enabled and is polling the CUCM to obtain the latest phone registration status information for phone instances stored in the Automate database. The polling default interval is 43200 seconds (12 hours).

For details, see the *System Monitoring Configuration* (Metrics Collection) topic in the Advanced Configuration Guide.

However, When viewing a list of phones, the **status** action can be carried out by an administrator who has been assigned a role that has an access profile to enable this action. By default, an administrator *above* provider level can carry out this task via the **Access Profiles** page - in this case, for relation/SubscriberPhone.

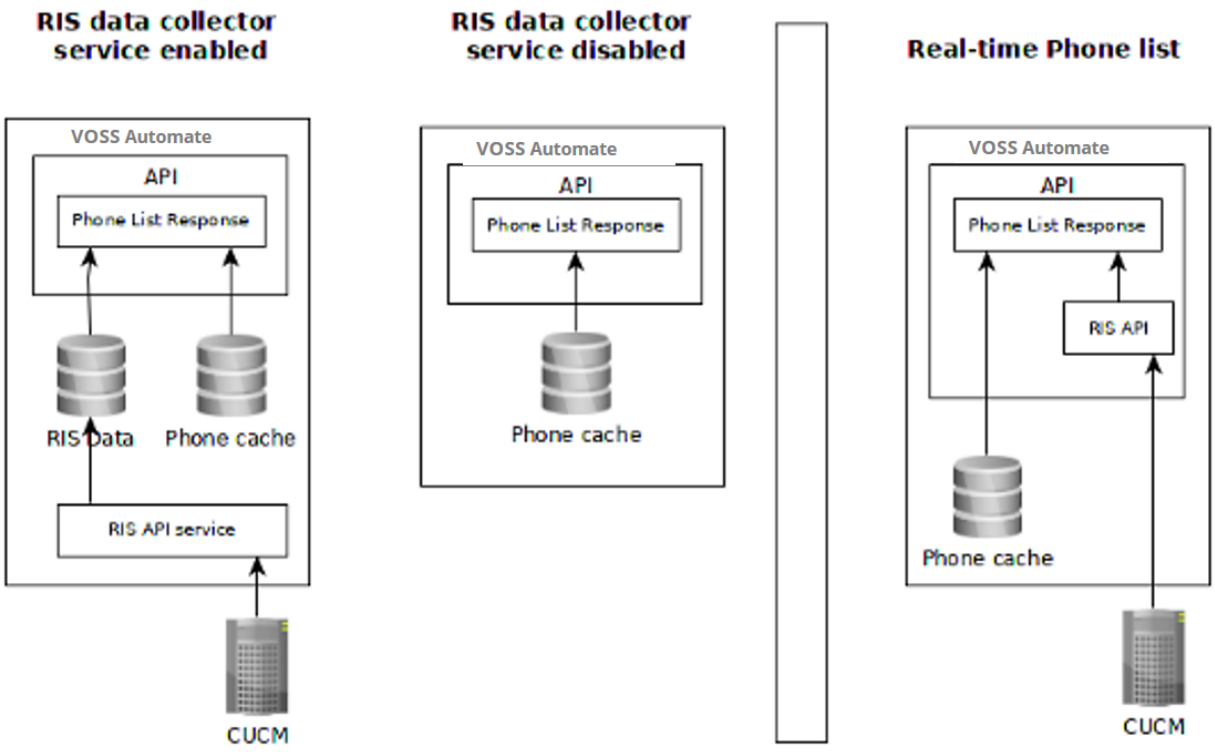
Carrying out this operation fetches the UCM phone IP address and status *directly* from the UCM and displays the data on the **Phones** list view **Registration Status** and **IP Address** columns, updating any existing data shown.

Important: Since the result of the **status** action is in real time, the current status of the list requires that the action carried out in order to see the latest values.

There is no caching of data resulting from this action. If any values show in the columns before carrying out this action, these would not be current, but are the cached values from the RIS data collector if it is enabled.

When carrying out the **status** action, the data in the **Registration Status** and **IP Address** columns can only be viewed.:

- The latest data only shows for the *current* list of phones on the GUI.
- The data in these columns is not stored in the database and *cannot be exported*.



Note:

- Whether the real-time information (RIS) data collector service is enabled or disabled, if the **status** action is carried out from the phones list view, the operation will *always* fetch and display the current information for the displayed phones directly from the device.
- The administrator's access profile associated with the role needs to allow the administrator to carry out the **status** action.
- The carrier-integrated Mobile device type is automatically added to the **RIS API Excluded Device Types** and therefore not fetched by the service.

When clearing or enabling the check box on `data/Settings`, log in on the platform command line interface (CLI) and restart the service:

```
$ cluster run application app start voss-risapi_collector
```

Phone status service in the logs

When clearing this setting and then restarting the (RIS) data collector service, an `app.log` entry will show: `"message":"RIS API service disabled"`.

Refer to the Platform Guide for commands to inspect log files.

Example log entry below (line breaks added):

```
2020-03-26T20:06:00.346577+00:00 VOSS-UN-2 deviceapi.background.risapi INFO
 {"process_id":24,
  "hostname":"VOSS-UN-2-voss-risapi-collector",
  "name":"deviceapi.background.risapi",
  "level":"INFO",
  "utc_iso_timestamp":"2020-03-26T22:06:00.346268",
  "request_uuid":null,
  "user_hierarchy":null,
  "user":null,
  "message":"RIS API service disabled",
  "line":330,
  "parent_process_id":1
 }
```

2.4.6. Additional role access profile validation

High level admins (`sysadmin`) with access to the `data/Settings` model (**Settings** page) can view and update the **Additional Role Access Profile Validation** setting to manage the available roles when an administrator creates access profiles.

The table describes how the system works when the **Additional Role Access Profile Validation** setting is enabled or disabled:

Enabled	An admin can only assign a role to a user if it is linked to an access profile with permissions that are in the subset of the admin's own access profile. Role drop-down lists will therefore be restricted. If the macro function `fn.filter_roles_by_user_access_profile` is used, the setting needs to be enabled for roles to be filtered. This validation check also applies when admins manage multi-role admin users - where the role is associated with an Authorized Admin Hierarchy.
Disabled	(Default) An admin can assign any role to a user, regardless of the admins own access profile. Role drop-down lists will therefore not be restricted. If the macro function `fn.filter_roles_by_user_access_profile` is used, the roles will *not* be filtered. This validation check also applies when administrators manage multi-role admin users - where the role is associated with an Authorized Admin Hierarchy.

Note: See the macro topic Filter Role Functions for details on the use of the `fn.filter_roles_by_user_access_profile` function.

2.4.7. File upload limitations

High level admins (`sysadmin`) with access to the `data/Settings` model (**Settings** page) can view and update file upload size limitations and file time to live on the database.

By default, the following file limitations apply:

- Maximum Upload File Size: 209715200 bytes (200MB)
- Time-To-Live for Uploaded Files: 24 (clean up every 24 hours)

Note:

- Extending the Maximum Upload File Size to greater than the default can impact the platform system operation.
- The minimum setting for time-to-live hours is 1 hour

Files are uploaded to the system database during activities such as:

- Bulk Load
- JSON import
- Theme upload
- Any other file upload activity, *excluding*:
  - Themes
  - SSO certificates
  - SSO service provider metadata
  - Audio files (MoH)

The time-to-live value applies to uploaded files that have not been used, in other words, imported or processed. By default, a check is done every 24 hours for such files, after which time they are removed.

2.4.8. Time-to-live for uploaded files

High level admins (`sysadmin`) with access to the `data/Settings` model (**Settings** page) can view and update the time-to-live for uploaded files.

Note: Files uploaded from file management menus on the Automate GUI and listed as instances of `data/File` are not affected.

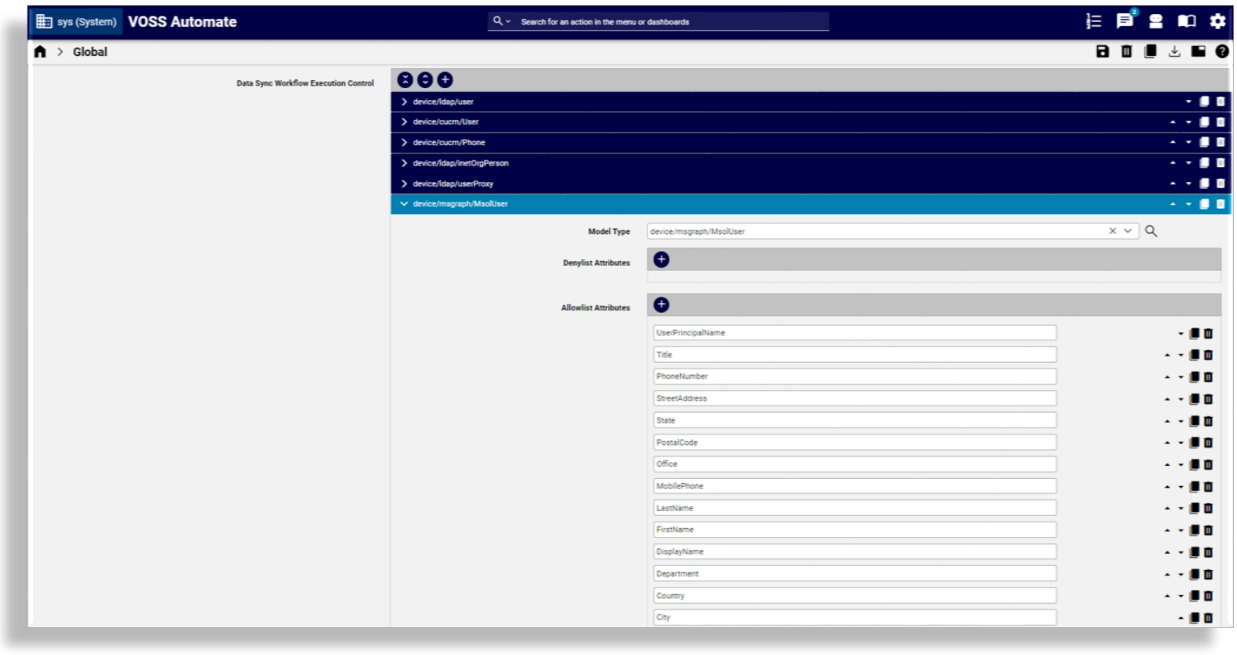
Uploaded bulk load files and imported JSON files are affected; however:

- For bulk load files, the file is kept for as long as there is an instance of `data/BulkLoad` attached to it. So a schedule that is more than 24 hours in the future is not impacted, because when we schedule a bulk load for the future, we create a `data/BulkLoad` instance. The instance is cleared when the bulk load is executed.
- Monthly license reports that are uploaded to the database by the internal schedule are not removed. For more details, refer to the License Guide.

2.4.9. Data sync workflow execution control

High level admins (`sysadmin`) with access to the `data/Settings` model (**Settings** page) can view and update lists of device attributes affected by data sync in the **Data Sync Workflow Execution Control** settings (allowlist and denylist attributes):

List	Description
Allowlist Attributes	When this list contains a field, then *only* a change in that field and not any other field will trigger data sync workflows, regardless of the list of the **Denylist Attributes**. In other words, this list takes precedence over the existing list of **Denylist Attributes**.
Denylist Attributes	Items in this list will *not* trigger any update workflows that may have been defined to execute during the data sync. These attributes are therefore excluded from data sync considerations. The reason for this list of attributes is that while data sync operations can have a performance impact, some data sync attribute changes do not require data sync workflows to be carried out. Note however that *the local device cache will still be updated with the updated attribute data*. No update workflows will be run, though. The transaction logs will indicate the updated device cache, but the transactions for these attributes instances will show as: *Device changes on denylisted attributes only. Updating cache, skipping workflows.*



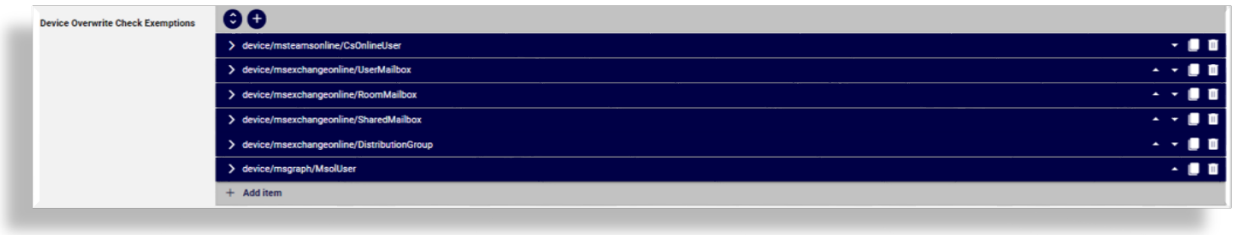
Related topics

- Allowlists and Denylists in the Core Feature Guide.
- Microsoft syncs in the Best Practices Guide.

2.4.10. Device overwrite check exemptions

High level admin users (`sysadmin`) with permissions to access the Global instance of the settings in the `data/Settings` model can create a list of fields to be excluded from device overwrite checking.

This means that if the field changes on the device, it will *not* be overwritten by data in Automate. The most common situation where this might be necessary is where a device field changes, but does not affect the data on Automate, because the data is treated as read only in Automate.

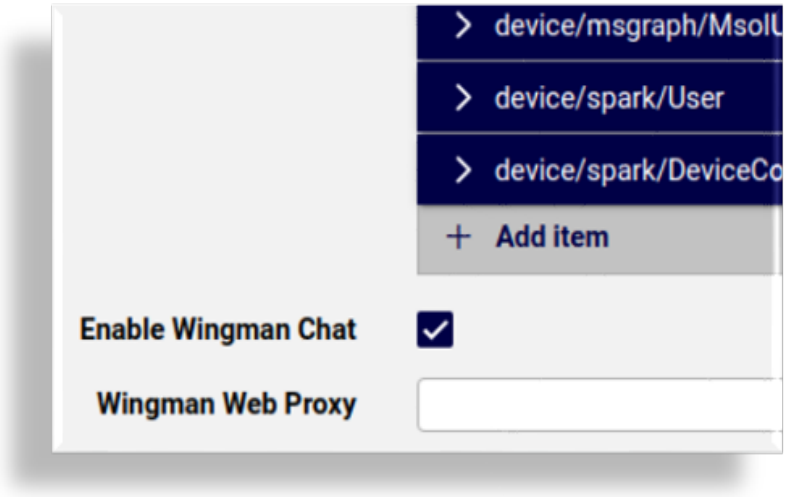


2.4.11. Enable Wingman chat

High level admin users (`sysadmin`) with permissions to access the *Global* instance of the settings in the `data/Settings` model (**Settings** page) can enable or disable the **VOSS Wingman** AI assistant or copilot.

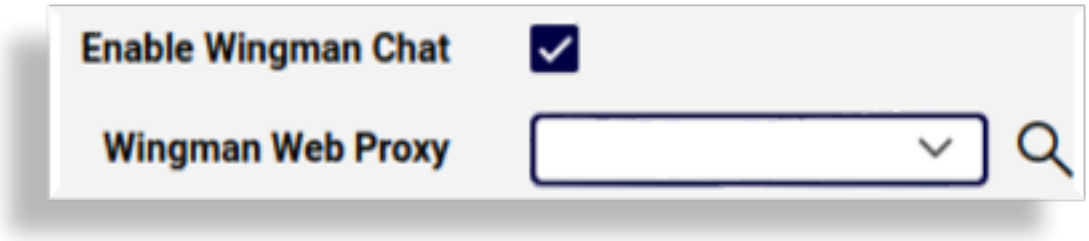
By default, **Enable Wingman Chat** is enabled, and the **Wingman** icon displays on the Admin Portal toolbar, if the user role is associated with an access profile that has **Wingman Chat** enabled under the **Miscellaneous Permissions**.

See *VOSS Wingman* in the Core Feature Guide.



2.4.12. Wingman web proxy

High level admin users (`sysadmin`) with permissions to access the *Global* instance of the settings in the `data/Settings` model (**Settings** page) can select a Web Proxy name that has been added as an instance on the **Web Proxy** menu and which will be used by VOSS Wingman on environments that require web proxy access to the internet.



Related topics

- "Set up a web proxy" in the Core Feature Guide.
- VOSS Wingman in the Core Feature Guide.

2.4.13. Portal Preferences

The **Portal Preferences** settings define system-wide settings for your environment.

The table describes the available settings:

Setting	Description
Application Insights	Enables/disables Automate product usage event tracking. Enabled by default. Clear the field to disable event tracking. When enabled, the system securely tracks and measures anonymized product usage events, such as login/logout actions, user hierarchy, platform ID, locality, and browser type. Additional tracked events include console errors, transaction types (add/update), transaction status (success/fail), and the names of dashboards, list views, and forms opened/closed. Event tracking in list views includes the number of results returned and filter operators used (for example, “contains”, “equals”), but not filter values. Wingman queries are excluded, with only the number of queries recorded. This data helps VOSS gain insights into product usage and improve user experience.
Tooltip Display Event	Defines the action (event) that allows tooltips to display. You can choose to have tooltips display for form elements only when using keyboard navigation (focus), or only with mouse-over (hover) over the form element, or for both hover and focus, or neither hover or focus (that is, no tooltips).

2.5. Global settings

Tip: *Use the Action search to navigate Automate*

2.5.1. Overview

Administrators (Provider level and up) may configure global settings for customizations that apply at a specific hierarchy only or across all hierarchies in the system.

On each tabbed page in Global Settings, a read-only field below the choice drop-down displays the current setting for your system. Options are:

Inherit	The service is enabled/disabled based on the setting at the hierarchy above the current one.
Yes	The service is enabled at the current hierarchy.
No	The service is disabled at the current hierarchy.

To change inherited settings, see *Changing inherited settings*.

2.5.2. Update Global Settings

Global settings are configured on the tabs of the Global Settings page:

- *Number Inventory*
- *Number Inventory Alerting*
- *Microsoft Licensing Alerting*
- *Webex App*
- Pexip Conference
- Email
- Phones
- Voicemail
- User
- Flow Through Provisioning
- Enabled Services
- *Enabled Solutions*

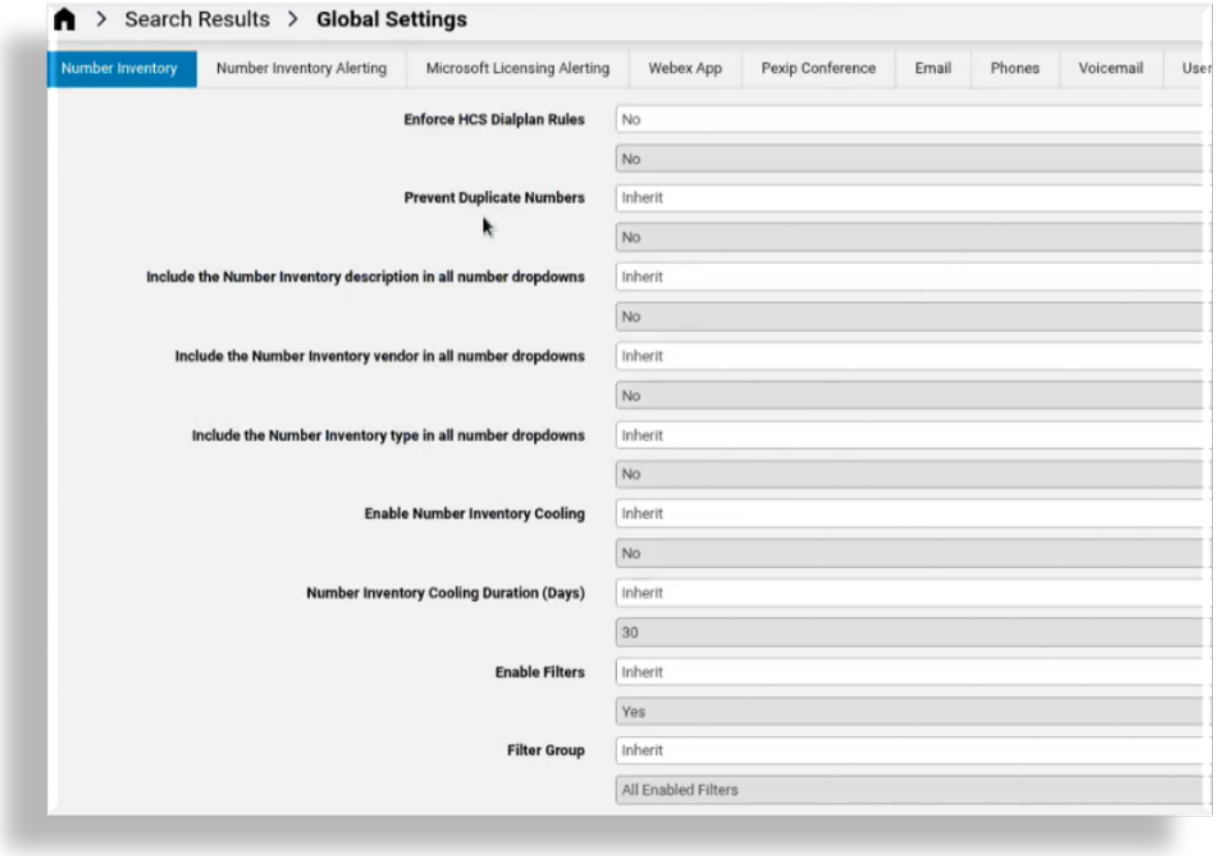
Number Inventory

The table describes the global settings for the Number Inventory:

Field	Description
Enforce HCS Dialplan Rules	When enabled, dial plan workflows enforce HCS Rules when provisioning Customers, Countries, Site and so on. Default = **Inherit**. If your deployment uses a custom or specific dial plan that does not conform to the HCS rules, this setting should be set to **No** (False).
Prevent Duplicate Numbers	This setting displays only at hierarchies above site or or linked site levels, and only when Microsoft is enabled for your system (via the Enabled Services tab in Global Setting). When available, the setting is enabled only when *Enforce HCS Dialplan Rules* is set to *No* (disabled), else, the field is read-only. In Microsoft environments, defines whether to allow the creation of duplicate numbers at sites in Automate when syncing in and provisioning users, or when creating number ranges. Default is *No* (duplicate numbers are allowed). The system checks the setting for *Enforce HCS Dialplan Rules* before applying the *Prevent Duplicate Numbers* logic. When enabled, the system enforces unique number validation throughout the hierarchy.
Include the Number Inventory description in all number dropdowns	Defines whether descriptions for the numbers (which can be added when the number inventory is managed via **Number Management**), display along with the numbers in the drop-down lists. For example, let's say you have a number and its description as follows: *1000 - CEO Internal*. When this setting is enabled (Yes), both the number (1000) and its description displays in the lists (when using features such as Quick Add User). The default is No.
Include the Number Inventory vendor in all number dropdowns	Defines whether vendor names for the numbers show in number dropdown as an option. For example, a number `982017206` (which is from Microsoft vendor) will display as `982017206 [Microsoft]` in the drop-down list.

Field	Description
Include the Number Inventory type in all number dropdowns	Defines whether number types for the numbers show in number drop-down as an option. For example, a number `982017206` (which is from Microsoft vendor and is of type `OperatorConnect` will display as `982017206 [OperatorConnect]` in the drop-down list.
Enable Number Inventory Cooling	Defines the availability of numbers in the system when a phone, user, or service associated with the number is deleted, and the number is no longer associated with these entities. Options are: • **Inherit**: When set to True, number inventory cooling is enabled or disabled based on the setting defined for number inventory cooling at a higher level in the hierarchy. • **Yes** (True) Enabled at the current hierarchy. Numbers associated with deleted entities are kept in a cooled state for a specified number of days (based on the value defined in the **Number Inventory Cooling Duration (Days)** field. Numbers in a cooled state are unavailable in the system until the cooling period end date is reached, unless they are manually released before the "end cooling period" end date. • **False** (No) Default. Number inventory cooling is disabled by default.
Number Inventory Cooling Duration (Days)	When number inventory cooling is enabled (Yes/True), this field defines the period (number of days) the number is kept in a cooled state and unavailable for association with a phone user, or service. The default is 30 days.
Enable Filters	Enables/disables custom number inventory filters at the current hierarchy. Shipped inventory filters can't be enabled or disabled as these reside at *sys* level. When enabled, custom and shipped inventory filters display in a drop-down list on forms such as Quick Add User, Onboard User (to add a user from a profile), Multi vendor service user move, Cisco Advanced User, Phones, Extension Mobility, and Single Number Reach. Options are: • **Inherit** (default): When set to True, custom inventory filters are enabled or disabled based on the setting defined at a higher level in the hierarchy. • **Yes** (True) Enables custom inventory filters at the hierarchy. When set to True, select the inventory filters to make available for selection in relevant drop-down lists, either shipped and custom inventory filters or all enabled inventory filters. You can view the available filters at a hierarchy via **Manage Filters**. When enabled, you can enable or disable specific filters at specific hierarchies via **Manage Filters**. • **False** (No) Custom and/or cloned shipped filters are disabled at the hierarchy.

Field	Description
Filter Group	When **Enable Filters** is set to Yes, the selected filter defines the number inventory filters available in the the drop-downs for choosing a number inventory filter. Options are: • Inherit • Shipped Enabled Filters • Custom Enabled Filters • All Enabled Filters (default) When set to *Inherit*, the value displays for the filter group that will be used from a higher level in the hierarchy. Custom enabled filters will only display in the drop-downs if these exist and are enabled at the hierarchy you're working at. *All Enabled Filters* includes all custom and shipped filters, provided they are enabled at the hierarchy you're working at.



Related topics

- Number Cooling in the Core Feature Guide
- Number Cooling Auto Expiry Schedule in the Advanced Configuration Guide
- Manage Number Filters in the Core Feature Guide
- Prevent duplicate numbers in the Core Feature Guide

Number Inventory Alerting

This tab configures the global settings for number inventory alerting, which defines how alerts may be raised once the number inventory is running low.

The table describes the settings on this tab:

Field	Description
Enable Alert on Available Numbers	By default, this setting is set to **Inherit**. However, it will not inherit the setting from higher up the tree unless it is explicitly set to **Yes** or **No**. **Inherit** in this instance just means it is *not configured*. Change to **Yes** to enable alerting.
Alert Aggregate Level	Choose a hierarchy level at which the *aggregate* of available numbers should be calculated (Provider, Reseller, Customer, Site), and displayed in the body of the alert. The shown data in the body for this hierarchy level is: • Hierarchy node name • Hierarchy node type • Hierarchy full path • Total numbers available • Total numbers • Total percent available Data is also included for lower hierarchies (as tables and in CSV format). For details, see the following topics: • Email in the Core Feature Guide • Number Inventory Alerts in the Core Feature Guide
Availability Threshold Percentage	Select or enter a percentage available of the total numbers at which point alerts will be raised. Sample percentages are available to choose from. If available numbers drop below this percentage, alerts will be raised.
Enable Email Group	Set to **Yes** to send email alert notifications to an email group. The email group needs to be available or should be set up.
Alert Email Group	If **Enable Email Group** is set to **Yes**, select the email group.
Ignore Hierarchies With No Numbers	If set to **Yes**, hierarchy levels with no numbers are excluded from reports.

Note: The email alert message also includes an attachment file `NumberThreshold.csv`, which contains the alert report in CSV format. See: *Email HTML templates*.

Related topics

- Email in the Core Feature Guide
- Number Inventory Alerts in the Core Feature Guide
- SNMP Traps: Number Inventory Alerting in the Platform Guide

Microsoft Licensing Alerting

The table describes the global settings for the Microsoft Licensing Alerting:

Field	Description
Enable Alert on Microsoft Licenses	When enabled, alerts will be raised and optionally emailed if the Microsoft license availability threshold is reached at the hierarchy. Default = **Inherit**. Note that this setting requires that **Enable Microsoft User License Enforcement** is enabled under the **User** tab. Refer to **Availability Threshold Percentage** below.
Availability Threshold Percentage	Defines a percentage of remaining Microsoft licenses at which an alert is raised. Percentages are available in the drop-down lists: 10, 15, 20, 25. Default = **Inherit**.
Enable Email Group	Defines whether an email group will receive alerts. Default = **Inherit**. If set to **Yes** at the hierarchy, the **Alert Email Group** drop-down list provides available email groups.
Alert Email Group	The selected email group to receive alerts for Microsoft licenses consumption above the defined threshold. Default = **Inherit**.

Related topics

- Microsoft License Management and Alerting in the Core Feature Guide
- Email in the Core Feature Guide

Webex App

This tab configures the global settings for Webex App.

The table describes the fields on this tab:

Field	Description
Retain a Webex App User when a subscriber is deleted	Defines whether to delete Webex App user when a user is deleted. Default is **No**.
Send notification when the Webex App Refresh Token expires	Defines whether a notification is sent when the Webex App refresh token expires for a specified customer. A SNMP trap and Webex App message is sent to recipients configured in the email group.
Webex App Refresh Token expires threshold (in seconds)	The max threshold (in seconds) for when to send a SNMP trap to the SNMP (if **Send SNMP trap message when the Webex App Refresh Token expires** is enabled). The default is `172800` seconds, which is two days.
Automatically apply default calling behavior on Webex App user data sync	Whether to apply default calling behavior (set up in Customer settings), to new Webex App users synced in to Automate. Default is No.
Generate and send Webex App User CSV file via Webex App message	Whether to generate a CSV file on create/update of Webex App user. Default is No. If enabled (Yes), the CSV file is sent via Webex App to a predefined list of recipients.
Email group containing recipients of the generated Webex App user CSV file	The group of recipients of the Webex App message with the generated CSV file. The email group is set up on the **Email Groups** menu.
Send manual Webex App Workspace configuration steps via Webex App message	Whether manual configuration steps (on Webex App Control Hub) are to be sent on create/update of a Webex App workspace. Default is **No**. If enabled, the steps are sent via a Webex App message.
Email group containing recipients of the manual Control Hub steps	Email group recipients of the Webex App message containing the manual configuration steps.
Quick Add Group for Hybrid Calling Workspace Unified CM users	The Quick Add Group to use when creating dummy CUCM users with line and device for Webex App workspace hybrid calling.
Enable Cisco Webex Contact Center Model References	Defines whether to enable retrieval and display of Cisco Webex Contact Center device model references from the Webex Control Hub. It is recommended that you enable this setting in the Global Settings only for any customer where you want to retrieve the reference details. This is to prevent a performance impact on customers where the setting is not required.

Related topics

- Quick Add Groups in the Core Feature Guide
- Email in the Core Feature Guide
- Email Groups in the Core Feature Guide
- Create Webex App Service in the Core Feature Guide

Pexip Conference

This tab configures the global settings for Pexip Conference.

The table describes the settings on this page:

Field	Description
Retain a Pexip Conference when a user is deleted	Defines whether the Pexip conference set up from the user interface is to be removed when the user is deleted. By default the setting is inherited from the hierarchy level directly above the current one.

Email

This tab configures the global settings for Email.

The table describes the settings on this page:

Field	Description
Allow welcome email to be sent to user after Quick Add User	Defines whether an email is sent to a user when added via Quick Add User. The default is **No**. When set to **Yes**, and a SMTP server is set up via **Apps Management**), then selecting the option to send an email when using Quick Add User, a welcome email is sent to the new user.

Related topics

- SMPT Server in the Core Feature Guide
- Email in the Core Feature Guide

Phones

This tab configures the global settings of phones for a site.

Note: These settings only apply to phones within the same site; both the re-added phone and the existing phone must be on the same site.

The table describes the phone global settings on this tab:

Field	Description
Delete existing Unassigned Phone when re-adding an identical phone	Defines whether to delete an existing, unassigned phone (a phone without an owner), when re-adding a phone with the same name and product type (duplicate phone). Default is *Inherit* (*No*, inherited from the hierarchy above), which triggers a transaction failure if you try adding a duplicate phone, for example, in a Quick Add User bulk load or when updating a user. When set to *Yes*, a system check verifies whether the phone exists and/or if it is already assigned (whether ownerUserName field is populated): • If the phone exists and is assigned to a different user, the transaction fails. • If the phone exists and is unassigned, the existing phone is deleted, the phone is re-added, and is assigned to the user you're adding or updating. • If the phone exists and is already assigned to the user you're working with. The system performs an update.
Retain Desk Phones when Subscriber is deleted	Defines whether a user's associated desk (hard) phones (phones prefixed with SEP or BAP) are deleted or retained when that user is deleted. When set to *Yes*: • The deleted user's hard phones are retained. • The deleted user's soft phones (such as Jabber devices) are deleted. • An additional field displays (**Update the Retained Desk Phone with Configuration Template**), which allows you to define whether retained phones are updated via a CFT once the user is deleted. Default is Inherit (set to No). This setting defines hard phone delete/retain behavior for any method of deleting a user, for example, delete user via the User's list view, or delete user in LDAP import, purge or sync (where delete or purge mode is automatic). You can view the hard phones associated with the user on the Phones tab in the user settings.
Update the Retained Desk Phone with Configuration Template	This field displays only when **Retain Desk Phones when user is deleted** is set to **Yes** (True). Defines whether to update retained hard phones via a configuration template (CFT) when the associated user is deleted. This feature ships with a default CFT (RemoveOwnerFromPhoneCFT), which clears the phone's Owner User ID if the phone is retained when deleting the associated user. You can choose a different CFT for the update step, if required.

Field	Description
Include additional information in Phone dropdowns	Options are Yes, No, Inherit. Default is *Inherit* (inherited from the hierarchy above) Set to *Yes* to enable. You will need to save this update then refresh the tab to display an additional configuration field (Additional information in Phone dropdowns).
Additional information in Phone dropdowns	Options are Yes, No, Inherit. Default is *Inherit*. Additional information options are *Description*, *First Line*, and *Description + First Line*. The default, *Description*, means that the phone description (if defined) displays in the phone selection drop-downs on the **Replace Phone** configuration page (Existing Phone tab, Device Name drop-down), and on Quick Add User, allowing you to search by phone description when choosing a phone from the drop-down. In the same way, when the additional information option is set to either *First Line* or *Description + First Line*, you can search for or choose phones based on this criteria.
Prevent Duplicate MAC Addresses for Cisco Phones	Options are Yes, No, Inherit. Default is *Inherit*. For any transaction adding a Cisco phone, if this setting is enabled, the transaction will fail with a message: `Phone already exists with name:` if any phone is found containing the same MAC address within all clusters in a customer or reseller.

Related topics

- Replace Phone in the Core Feature Guide.
- Quick Add User for Cisco UCM Users in the Core Feature Guide.
- User, Phones tab in the Core Feature Guide

Voicemail

This tab configures the global settings for voicemail.

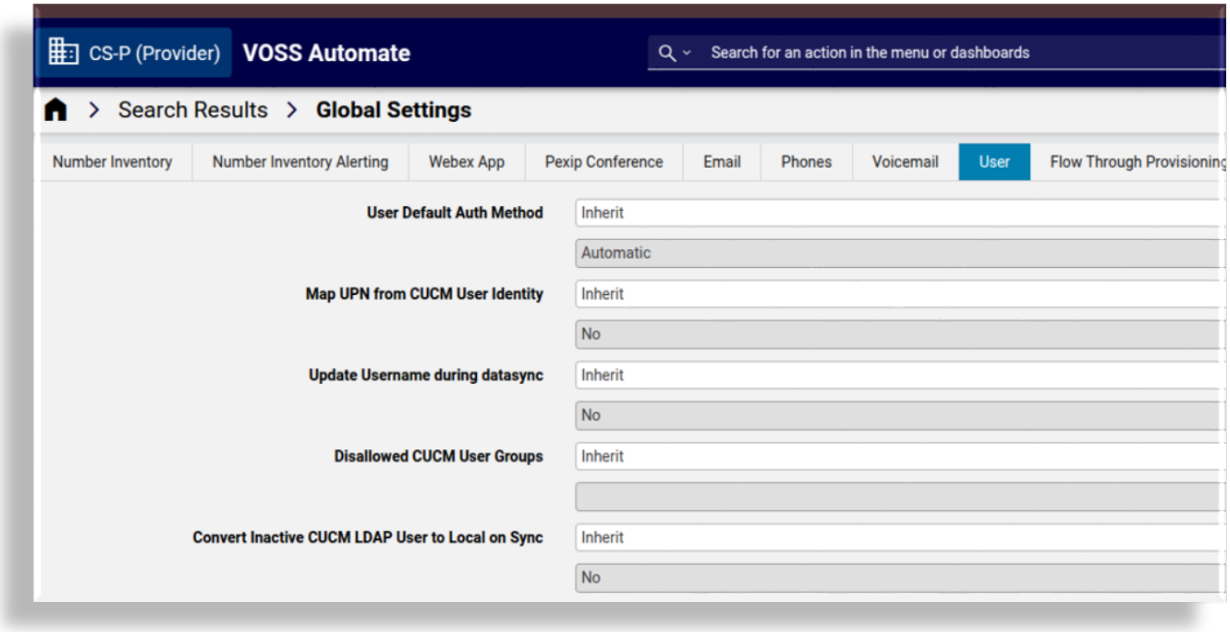
The table describes the settings on this tab:

Field	Description
Retain a (Cisco) Voicemail Account when a user is deleted by data sync only	Defines whether to retain a Cisco (UCM) user's voicemail account when the user is deleted. Default is Yes (true). When set to Yes, the CUCM user's voicemail account is retained when the user is deleted and user sync is executed.

User

This tab configures the global settings for users.

Note: When a user is either synced into or added manually on Automate, these settings apply by default. The settings can however be modified when adding a user via **User Management**.



The table describes the settings on this tab:

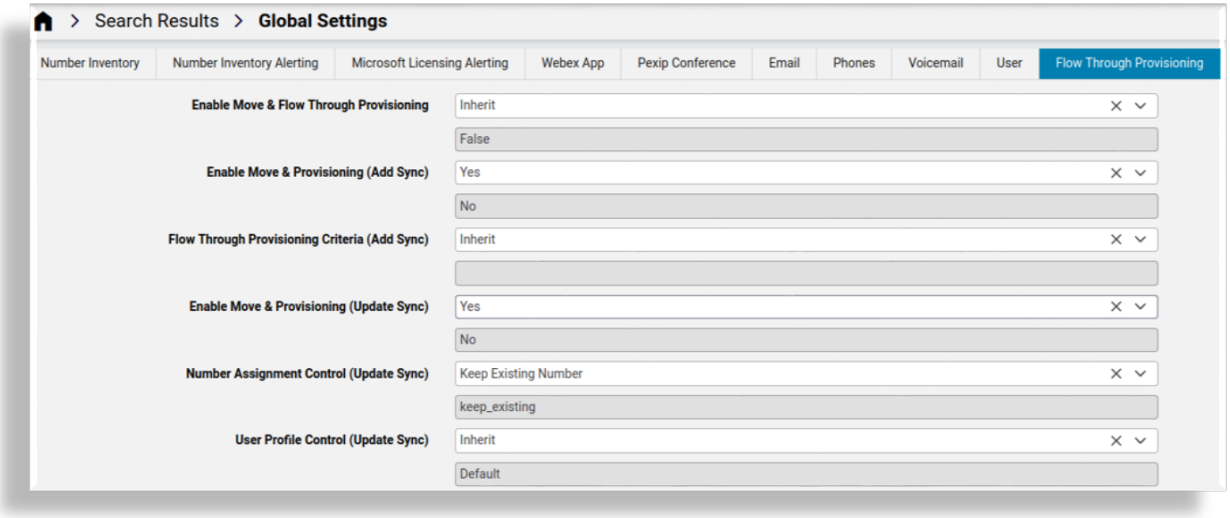
Field	Description
User Default Auth Method	The default authentication method to use when a user is synced in or added manually. The default is **Local** (inherited).
Map UPN from CUCM User Identity	Maps the Microsoft Azure UserPrincipalName (UPN) attribute to CUCM userIdentity attribute - used in Cisco-Microsoft hybrid configurations where the same user ID is on every user account (MS Teams, CUCM, etc.). If enabled, the CUCM user's userIdentity attribute is used for the import of MS teams CsOnlineUser and MS 365 Msol user instances. The default is **No** (inherited).
Update Username during datasync	Defines whether to update the existing VOSS username when a new associated user is imported via a sync.
Disallowed CUCM User Groups	Defines the user groups (one or more) that admins will not be allowed to assign to user. This is to prevent users being incorrectly assigned elevated permissions to system resources that are reserved for users in the groups you specify here. Fill out the user group names in a colon-separated format, for example, *Standard CUCM SuperUser:MyGroupName*
Enable Microsoft User License Enforcement	Defines whether to Microsoft license allocation is enforced at hierarchy levels. Refer to the related topic below. Default is **Inherit**.
Convert Inactive CUCM LDAP User to Local on Sync	Defines whether to convert CUCM users that would normally be automatically deleted by the CUCM, to be converted into CUCM Local users during a data sync of CUCM. When disabled (default), users that have been in status "Inactive LDAP Synchronized User" for more than 24 hours are automatically deleted by the CUCM. These users and their services are then deleted from Automate on the next CUCM data sync. When enabled, users that have changed their status to "Inactive LDAP Synchronized User" are converted to "Enabled Local Users" on the next CUCM data sync. The data sync of the CUCM must occur within 24 hours of the users becoming inactive otherwise, CUCM will still delete them.
Retain User at Site after MS Offboard User	Defines whether to retain a user at a site instead of moving the user back to customer level when performing the task: Quick Offboard User. The default is **No** (inherited).

Related topics

- User Authentication Methods in the Core Feature Guide.
- Microsoft License Management and Alerting in the Core Feature Guide
- Convert user type CUCM-LDAP to CUCM Local in the Core Feature Guide.
- Microsoft User Management, Offboard User in the Core Feature Guide.

Flow Through Provisioning

This tab defines global settings for sync with flow through provisioning.



The table describes the settings on the Flow Through Provisioning tab/panel:

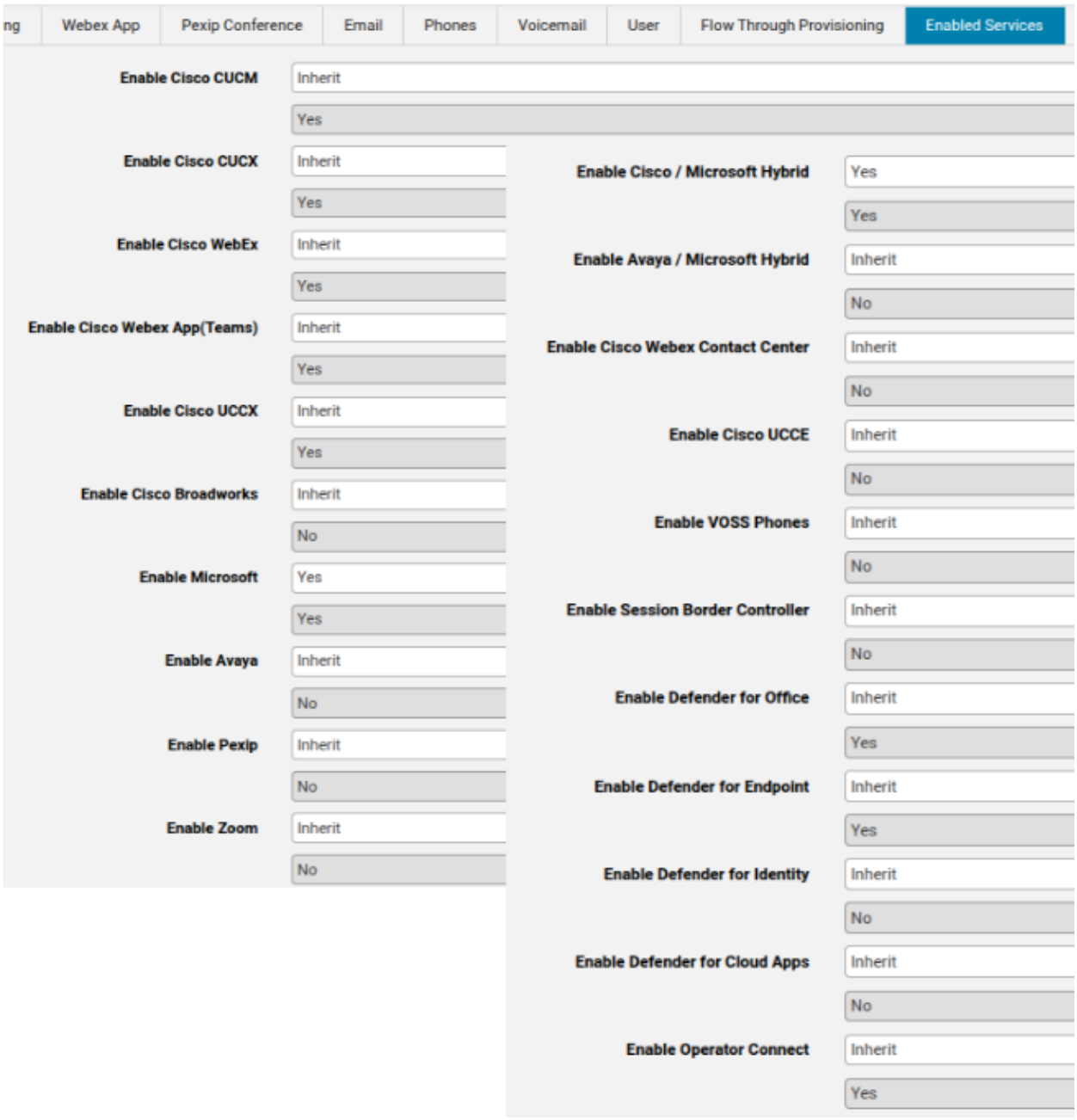
Field	Description
Enable Move & Flow Through Provisioning	Defines whether move and flow through provisioning is enabled. The default is No.
Enable Move & Provisioning (Add Sync)	Enabled only when **Enable Move & Flow Through Provisioning** is enabled. Defines whether move and flow through provisioning on *Add* sync is enabled. The default is No.
Flow Through Provisioning Criteria (Add Sync)	Enabled only when **Enable Move & Flow Through Provisioning** is enabled. Used *only* for adding and onboarding users. Defines the default criteria applied in an *Add* sync when moving and provisioning a user with flow through provisioning. The user is synced in, moved to a site, and provisioned with relevant serves in one step, based on the flow through provisioning criteria.
Enable Move & Provisioning (Update Sync)	Microsoft users only. Allows the system to automatically move an existing non-hybrid, Microsoft-only user, from one site to a new site, with their services and a new line, in a scheduled or manually triggered sync. When set to Yes, triggers display of these additional fields to apply controls for move using between sites, and provisioning in an update sync: • Number Assignment Control (Update Sync) • User Profile Control (Update Sync) **Important:** Update syncs for move and flow through provisioning execute *only* on users that are already at Site level. These are users who have already been provisioned at a site and their settings on Azure have changed, therefore requiring a move to another site and, if relevant, changes to their provisioning at the new site. Users synced in at Customer level as Msol users aren't eligible for flow through provisioning. To move and automatically provision these users you can either purge them and sync them in again in an Add sync with flow through provisioning, or manually configure and move them.
Number Assignment Control (Update Sync)	Displays only when **Enable Move & Provisioning after Update Sync** is set to Yes. Update sync only. Defines whether a user keeps their existing number or is assigned a new number when moved from one site to another site. Options are *Inherit* (from the hierarchy settings above this one), *Keep Existing Number* (default), or *Assign New Number*. Note that this is a global setting at the hierarchy where you're applying the setting so all users updated in this move and sync will either inherit settings, or keep their numbers, or be assigned new numbers.
User Profile Control (Update Sync)	Displays only when **Enable Move & Provisioning after Update Sync** is set to Yes. Update sync only. A single default profile used *only* in update syncs with flow through provisioning to apply criteria for moving the user. In an update sync, only this user profile applies for the move. The **Flow Through Provisioning Criteria** setting on this tab/panel applies only for *add* syncs.

Related topics

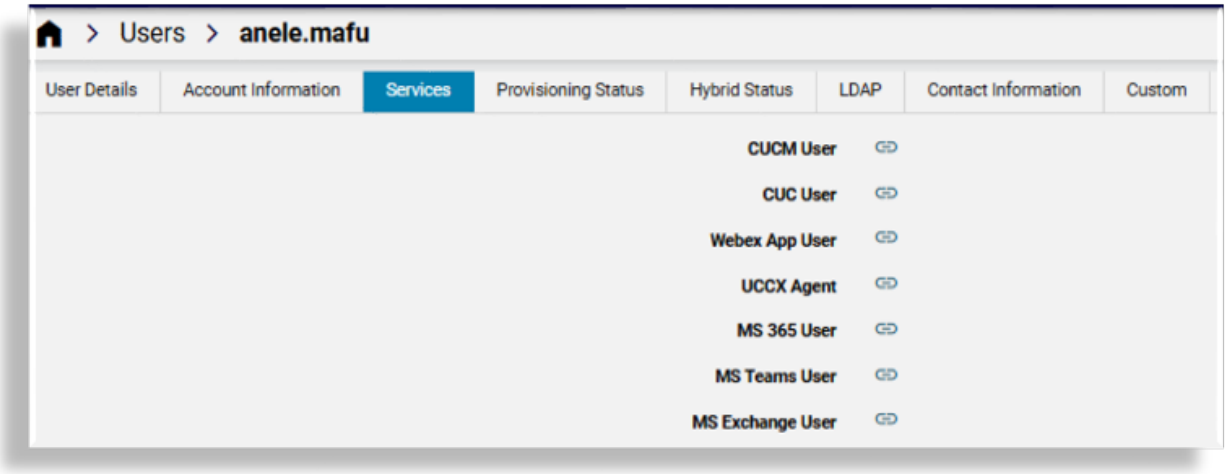
- Flow through provisioning (FTP) in the Core Feature Guide.
- Move Microsoft user and services in the Core Feature Guide.
- User profiles in the Core Feature Guide

Enabled Services

This tab defines the global settings for enabling/disabling services for different vendors, such as Cisco or Microsoft. Options are Inherit, or Yes/No (True/False).



Services that are available to users and enabled on this tab display on the **Services** tab of the user's management page. When enabled, an admin can click the link to the service details to view and update the settings for that service. For services that the user isn't using, you can disable the service (select *No*) on this tab so that it won't display on their user management **Services** tab/panel.



When provisioning services from two or more vendors, the global setting is the first of a number of system verification checks. For example, when the **Enable Cisco CUCM** global setting is set to **Yes** (enabled), the administrator can provision a user with new CUCM services (such as a Cisco phone, Jabber, and extension mobility), only if the CUCM device check (server installed), entitlement profile check, and field display policy check all pass the verification check. In the same way, if for example, the **Enable Microsoft** global setting is set to **No** (disabled), and all other checks are set to enabled, existing Microsoft services can be viewed but new Microsoft services cannot be provisioned.

Note: By default, for new installs, the global setting for the following services are inherited from higher levels in the hierarchy (Inherit set to True/enabled):

- Cisco CUCM
- Cisco CUCX
- Cisco WebEx
- Cisco Webex App
- Cisco CCX

When upgrading to a version of the system that allows multi vendor and hybrid users, the default setting for services other than these 5 services is *Inherit* (False).

To provision services to new users (added after an upgrade), you will need to enable the vendor service in global settings.

The table describes services that can be enabled/disabled on this tab:

Setting	Description
Enable Cisco CUCM	Enables/disables Cisco CUCM services. The default is *Yes*. When set to *Yes*, allows an admin user to provision a user with new CUCM services, such as a Cisco phone, Jabber, and Extension Mobility, provided the server is installed, and the entitlement profile and field display policy pass a verification check.
Enable Cisco CUCX	Enables Cisco CUCX (Unity) services. The default is *Yes*.
Enable Cisco WebEx	Enables/disables Cisco WebEx services. The default is *No*.
Enable Cisco Webex App (Teams)	Enables/disables Cisco WebEx App (Teams) services. The default is *No*.
Enable Cisco UCCX	Enables/disables Cisco UCCX (Contact Center Express) services. The default is *No*.
Enable Cisco Broadworks	Enables/disables Cisco Broadworks services. The default is *No*.
Enable Microsoft	When enabled, allows provisioning of Microsoft services. The default is *No*.
Enable Avaya	Enables/disables Avaya services. The default is *No*.
Enable Pexip	Enables/disables Pexip services. The default is *No*.
Enable Zoom	Enables/disables Zoom services. The default is *No*.
Enable Cisco/Microsoft Hybrid	Enables/disables Cisco/Microsoft hybrid services. The default is *No*. When enabled, Automate allows for provisioning users and services from both Cisco and Microsoft devices, and makes available an admin user parent menu called **Hybrid Cisco-Microsoft Management**, and associated access profiles. For details, see *Hybrid Cisco-Microsoft Management* in the Core Feature Guide.
Enable Avaya/Microsoft Hybrid	Enables/disables Avaya/Microsoft hybrid services. The default is *No*.
Enable Cisco Webex Contact Center	Enables/disables Cisco Webex Contact Center services. The default is *No*.
Enable Cisco UCCE	Enables/disables Cisco UCCE services. The default is *No*.
Enable VOSS Phones	Enables/disables VOSS phones services. The default is *No*.
Enable Session Border Controller	Enables/disables Session Border Controller services. The default is *No*.
Enable Defender for Office	Enables Microsoft Defender for Office - security services for Office 365
Enable Defender for Endpoint	Enables Microsoft Defender for Endpoint - various devices
Enable Defender for Identity	Enables Microsoft Defender for Identity - for identity related threats
Enable Defender for Cloud Apps	Enables Microsoft Defender for Cloud - cloud security tools
Enable Operator Connect	Enables Microsoft Operator Connect for Providers in Automate. When set to Yes (enabled), displays Operator Connect settings in *Additional Apps*

Related topics

- Microsoft Defender for Office security management and policies

Enabled Solutions

This tab lists solutions that can be enabled in your system, provided you have the necessary licenses.

Note: As at the release of 25.3, the following solutions are available and enabled by default (provided you have the required licenses):

- **UC Automation**: Offers the same functionality as shipped in Automate pre-25.3.
- **Email**: Relevant for Microsoft customers using Automate for MS Exchange pre-25.3.

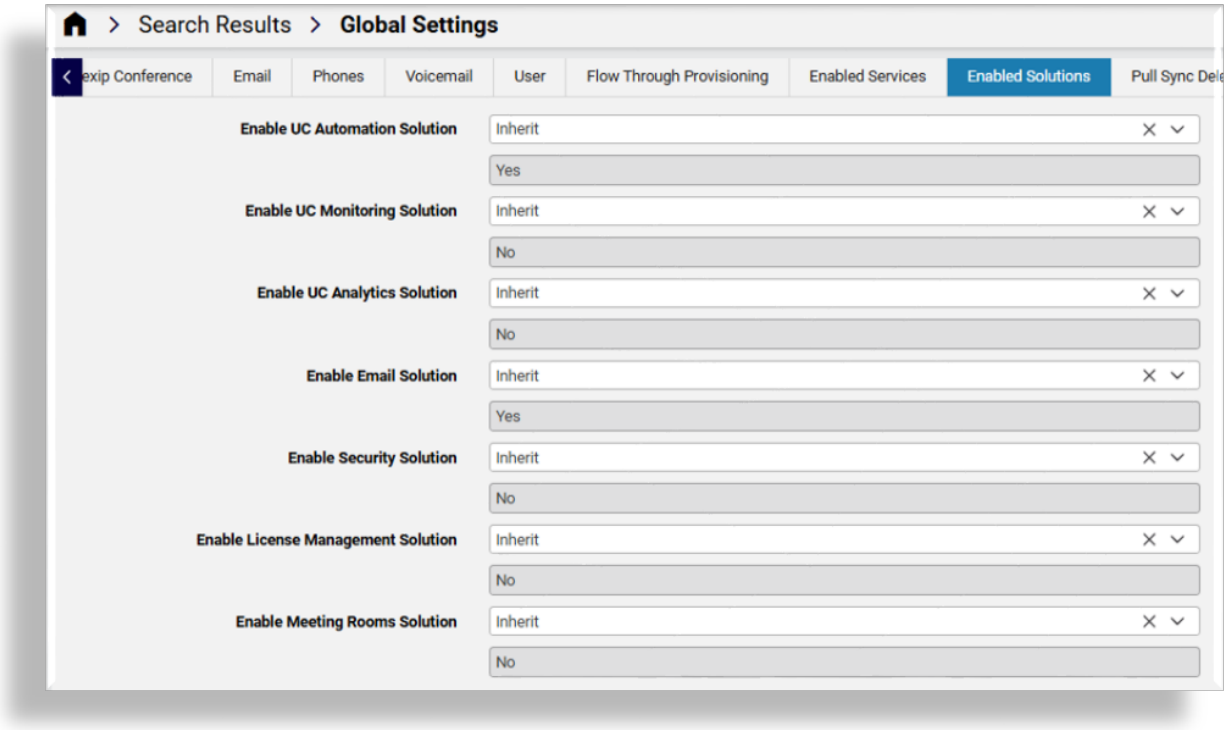
Deployment of the following solution capabilities is reserved for future development: UC Monitoring, UC Analytics, Security, License Management, Meetings Rooms

From 25.3, admins setting up a Microsoft tenant can add the tenant, select required solutions, and add the permissions for these solutions to a single application (app) registration directly in the customer tenant. When enabling two or more solutions, you will need to create your own app registration in this way, and add permissions to the app registration in the form of an uploaded certificate and/or a secret. Some solutions will require a certificate and others will need a secret.

Permissions required, by solution:

- UC Automation: Certificate
- Email: Certificate
- UC Monitoring: Secret
- UC Analytics: Secret
- Security: Certificate
- License Management: Certificate
- Meetings Rooms: Certificate

At the time of writing (25.3), shared central app registration can still be used provided you only have a single solution enabled; that is, *UC Automation* or *Email*.



Pull Sync Delete Thresholds

The Pull Sync Delete Threshold settings on the Enabled Services tab allow you to define the maximum number of items that may be deleted during a sync to protect against unwanted sync deletions. You can adjust the default values if needed. Sync will fail if the threshold is reached.

Setting	Description
Pull Sync Delete Threshold for CallManager	Blocks CallManager deletes if calculated number of local deletes exceeds the threshold. Default is 50.
Pull Sync Delete Threshold for LDAP	Blocks LDAP deletes if calculated number of local deletes exceeds the threshold. Default is 50.
Pull Sync Delete Threshold for MSExchangeOnline	Blocks MSExchangeOnline deletes if calculated number of local deletes exceeds the threshold. Default is 50.
Pull Sync Delete Threshold for MSGraph	Blocks MSGraph deletes if calculated number of local deletes exceeds the threshold. Default is 20.
Pull Sync Delete Threshold for MSTeamsOnline	Blocks MS-TeamsOnline deletes if calculated number of local deletes exceeds the threshold. Default is 20.
Pull Sync Delete Threshold for Spark	Blocks Spark deletes if calculated number of local deletes exceeds the threshold. Default is 50.
Pull Sync Delete Threshold for UnityConnection	Blocks UnityConnection deletes if calculated number of local deletes exceeds the threshold. Default is 50.
Pull Sync Delete Threshold for Zoom	Blocks Zoom deletes if calculated number of local deletes exceeds the threshold. Default is 50.

Related topics

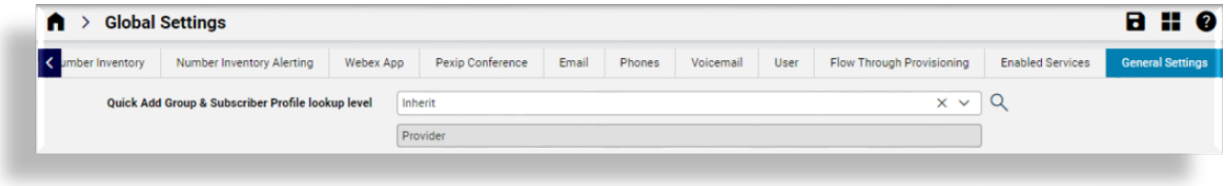
- Multi vendor users in the Core Feature Guide
- Role-based access for multi vendor users in the Core Feature Guide
- Configure multi vendor users in the Core Feature Guide
- Hybrid Cisco-Microsoft Management in the Core Feature Guide

General

This tab defines general global settings to manage system behavior.

The table describes services that can be enabled/disabled on this tab:

Setting	Description
Quick Add Group & User Profile lookup level	Specifies the hierarchy level up to which Quick Add Groups and user profiles will be searched for. The default is Provider level. (`sys` and `hcs` levels are not available.) When a lookup level is set, selections of available QAGs and user profiles will be restricted upwards to this lookup level. If you have hybrid customers (customers using both Cisco and Microsoft, for example), you can create hybrid-specific user profiles for those hybrid customers, then set the lookup level for those customers to *Customer* level so that they will have available a hybrid user profile in the drop-down at that customer hierarchy.



2.5.3. Changing inherited settings

- For numeric inherited values, for example, for “Number Inventory Cooling Duration (Days)” or “Webex App Refresh Token expires threshold (in seconds)”), you can overwrite the word “Inherit” with the required value, for example, 45, and save your changes. If the inherited value is already overwritten, for example, the value is already 45, then overwrite this value with the new value.
- For inherited values that are Yes/No (True/False), select an alternative from the drop-down (either Yes, No, or Inherit). This may change the current value.

2.6. Number cooling auto expiry schedule

The **Number Cooling Auto Expiry Schedule** (**IniCoolingExpiryCleanupService**) is a background process that runs daily to poll the **Cooling End Date** field on the Number Inventory list view, in order to determine which numbers have completed their cooling period.

Numbers that have reached their cooling end date are returned to the pool of available numbers at the specific hierarchy.

This process picks up any numbers with a cooling end date within the last 20 days. In this way, it can identify any numbers that reached their cooling end date while the system was down, or if the schedule was disabled for any reason.

Numbers with a cooling end date older than 20 days must be removed from cooling manually. A system administrator can manually release any cooled number, regardless of its cooling end date.

Important: An `hcsadmin` user (or higher) can view and manage the **Number Cooling Auto Expiry Schedule** (**IniCoolingExpiryCleanupService**) from the **Scheduling** list view.

While the `hcsadmin` user can edit this schedule, for example change the execution time or timezone, the schedule must run every day and should not be disabled.

Related topics

- Number Cooling in the Core Feature Guide
- Global Settings

2.7. Add a SMTP server

Tip: *Use the Action search to navigate Automate*

This procedure adds a SMTP server at a hierarchy level.

Prerequisites:

- Enable email in the Global Settings (Email tab).

Perform these steps:

1. Log in to the Admin Portal.
2. Choose the relevant hierarchy.

   Note: Configure the SMTP server at the hierarchy where you want to allow VOSS Automate to send email messages.

   You may only set up one SMTP server at each hierarchy level. The SMTP server will be available at the current hierarchy and below. For example, for a SMTP set up at a specific customer, the sites below that customer can use that SMTP server.

3. Go to **SMTP Server**.
4. Click the Plus icon (+) to add a new SMTP server.
5. On the **SMTP Server** page, fill out details for the new SMTP server:

Field	Description
Name	The SMTP server name.
Description	A description for the email account.
Port	The port number.
Secure	Relevant only for SSL connections to the SMTP server. Select the checkbox (enable) to use the SSL protocol for connections to the SMTP server. Default is disabled (checkbox is left clear), for TLS and unsecure logins to the SMTP server.
Username	The username credential for establishing a connection to the SMTP server.
Password	The password credential for establishing a connection to the SMTP server.

6. Save your changes.

Related topics

- Email in the Core Feature Guide
- Global Settings in the Core Feature Guide

2.8. Email

Tip: *Use the Action search to navigate Automate*

2.8.1. Overview

Provider administrators can test email messages and manage email templates, provided an email SMTP server is set up, and when emails are enabled via the **Email** tab of the *Global settings*.

Email functionality is available for the following:

Component	Description
Quick Add User - Cisco (QAS)	Enable email functionality via Global Settings > Email tab, then select a checkbox in QAS to send a welcome email to new users added via QAS.
Quick User - Microsoft	Enable email functionality via Global Settings > Email tab, then select a checkbox in Quick User to send a welcome email to new users added via Quick User.
File Transfer Destinations	Configured by high level system administrators to transfer audit data for licensing. See the Licensing and Data Export Guide.

Related topics

- *Add a SMTP server*
- *Global settings*

2.8.2. Send test email

On the **Send Test Email** page you can allow an email message to be sent to and from a specified email address, and select an email HTML template to test in the email body.

2.8.3. Email HTML templates

You can view and work with email templates on the **Email HTML Templates** page.

Email HTML templates contain placeholders for the email subject and body text, in HTML markup. The HTML markup can be:

- Previewed by using the **Preview** menu option in the editor
- Modified as required.

Default email templates

By default, the system provides the following email templates:

Note: When adding a HTML template from the list view, the **Name** can only be “Test Email Template”, “Quick Add User”, or “Number Inventory Alerting”.

Default email templates	Description
Test Email Template	This default template is read-only. You can't modify it or change its name. To use this template, you can clone it to your hierarchy and customize the clone.
Quick Add User	This default template is read-only. You can't modify it or change its name. To use this template, you can clone it to your hierarchy and customize the clone. You can use this template only if the setting is enabled via the Global Settings. Values from the Quick Add User input form can be used to populate the template by adding variables to the HTML template.
Number Inventory Alerting	This default template is read-only. You can't modify it or change its name. To use this template, you can clone it to your hierarchy and customize the clone. You can use this template only if the setting is enabled via the Global Settings. Values from Number Inventory Alert message can be used to populate the template by adding variables to the HTML template.

Quick Add User email template variables

Values from the **Quick Add User** page can be used to populate the Quick Add User email template by adding variables to the HTML template.

The table lists the available variables for the **Cisco** Quick Add User email template:

Field name on input form	Variable available in HTML
Username	`{{ pwf.EMAIL.username }}`
First name	`{{ pwf.EMAIL.firstname }}`
Last name	`{{ pwf.EMAIL.lastname }}`
One time password	`{{ pwf.EMAIL.password }}`
One time PIN	`{{ pwf.EMAIL.pin }}`
Access Code	`{{ pwf.EMAIL.phone_access_code }}`
Email	`{{ pwf.EMAIL.email }}`
Extension	`{{ pwf.EMAIL.extension_number }}`
Mobile Number	`{{ pwf.EMAIL.mobile_number }}`
Entitlement Profile	`{{ pwf.EMAIL.entitlement_profile }}`
Phone Type	`{{ pwf.EMAIL.phone_type }}`
Phone Names	`{{ pwf.EMAIL.phone_names }}`
Jabber Device Names	`{{ pwf.EMAIL.jabber_names }}`
Extension Mobility Name	`{{ pwf.EMAIL.extensionmobility_name }}`
External E.164 number	`{{ pwf.EMAIL.e164 }}`

Note: When sending the welcome email to users added via Quick Add User, if there is more than one E.164 number associated with the user's extension, only the primary E.164 number displays. If there are no E.164 numbers associated with the user's extension, then no E.164 number value displays.

The table describes the default variables for the **Microsoft** Quick user email template:

Field name on input form	Variable available in HTML
Username	{{ pwf.EMAIL.username }}
First name	{{ pwf.EMAIL.first_name }}
Last name	{{ pwf.EMAIL.last_name }}
One time password	{{ pwf.EMAIL.password }}
Email	{{ pwf.EMAIL.email }}
Extension	{{ pwf.EMAIL.line_uri }}
Mobile Number	{{ pwf.EMAIL.mobile_phone }}
Phone Number	{{ pwf.EMAIL.phone_number }}

Example user details you can add to your QAS HTML template:

```
<p>Username: {{ pwf.EMAIL.username }}</p>
<p>First name: {{ pwf.EMAIL.firstname }}</p>
<p>Last name: {{ pwf.EMAIL.lastname }}</p>
```

Number inventory alerting email template variables

Values from the Number Inventory Alert message can be used to populate the Number Inventory Alerting email template by adding variables to the HTML template. The table describes the variables available for this template:

Name on alert message	Variable available in HTML
Threshold of available (%)	{{ pwf.INI_ALERT_THRESHOLD }}
Threshold reached (True/False)	{{ pwf.INI_ALERT_THRESHOLD_REACHED }}
Hierarchy node type	{{ pwf.INI_ALERT_HIERARCHY_NODE_TYPE }}
Hierarchy friendly name	{{ pwf.INI_ALERT_HIERARCHY_NAME }}
Hierarchy full path	{{ pwf.INI_ALERT_HIERARCHY }}
Total Numbers Available	{{ pwf.INI_ALERT_TOTAL_INI_AVAILABLE }}
Total Number count	{{ pwf.INI_ALERT_TOTAL_INI_COUNT }}
Total percent available	{{ pwf.INI_ALERT_TOTAL_PERCENT_AVAILABLE }}
Table of usage per site	{{ pwf.INI_ALERT_NODES_EXCEEDED_THRESHOLD_TABLE }}

Example HTML

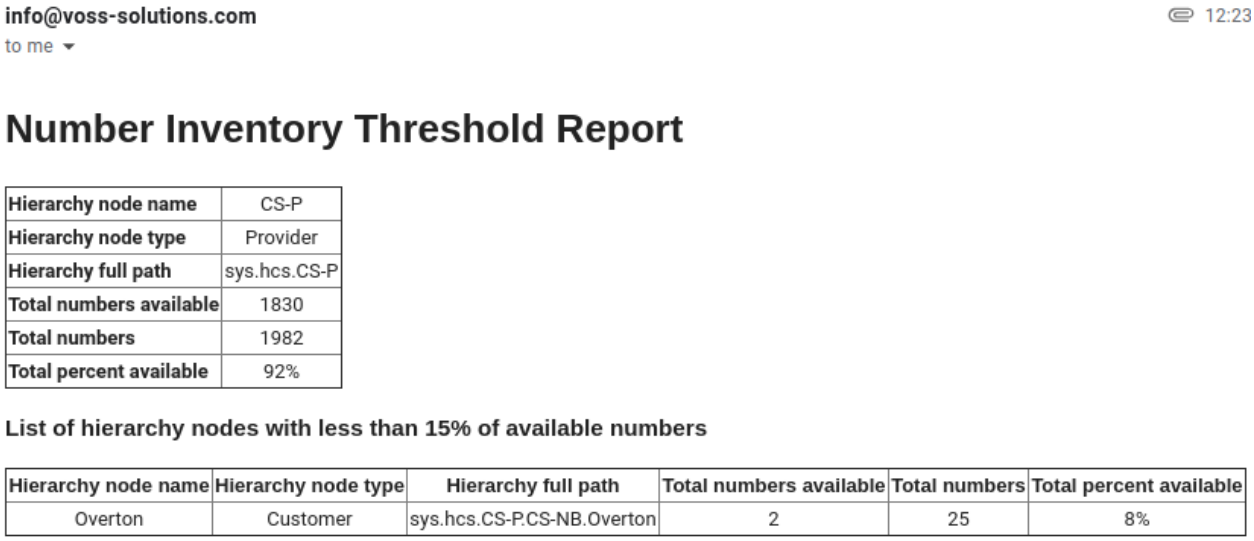
```
<h1>Number Inventory Threshold Report</h1>
<table border='1' style='border-collapse:collapse'>
<tr><td><b>Hierarchy node name</b></td><td><center>{{ pwf.INI_ALERT_HIERARCHY_NAME }}</
↪center></td></tr>
<tr><td><b>Hierarchy node type</b></td><td><center>{{ pwf.INI_ALERT_HIERARCHY_NODE_TYPE }
↪}</center></td></tr>
```

(continues on next page)

(continued from previous page)

```
<tr><td><b>Hierarchy full path</b></td><td><center>{{ pwf.INI_ALERT_HIERARCHY }}</center>
↪</td></tr>
<tr><td><b>Total numbers available</b></td><td><center>{{ pwf.INI_ALERT_TOTAL_INI_
↪AVAILABLE }}</center></td></tr>
<tr><td><b>Total numbers</b></td><td><center>{{ pwf.INI_ALERT_TOTAL_INI_COUNT }}</center>
↪</td></tr>
<tr><td><b>Total percent available</b></td><td><center>{{ pwf.INI_ALERT_TOTAL_PERCENT_
↪AVAILABLE }}%</center></td></tr>
</table>
<p></p>
<p>{{ pwf.INI_ALERT_NODES_EXCEEDED_THRESHOLD_TABLE }}</p>
```

Example message



info@voss-solutions.com 12:23

to me

Number Inventory Threshold Report

Hierarchy node name	CS-P
Hierarchy node type	Provider
Hierarchy full path	sys.hcs.CS-P
Total numbers available	1830
Total numbers	1982
Total percent available	92%

List of hierarchy nodes with less than 15% of available numbers

Hierarchy node name	Hierarchy node type	Hierarchy full path	Total numbers available	Total numbers	Total percent available
Overton	Customer	sys.hcs.CS-P.CS-NB.Overton	2	25	8%

The email alert message also includes an attachment file called `NumberThreshold.csv` that contains the alert report in CSV format, for example:

```
Hierarchy Node Name,Hierarchy Node Type,% Available,Total Numbers Available,Total Numbers
CS-P,Provider,92,1830,1982
CS-NB,Reseller,92,1830,1982
AAAGlobal,Customer,91,1428,1557
Overton,Customer,8,2,25
LOC001,Site,74,284,382
LOC002,Site,83,20,24
LOC003,Site,90,46,51
```

2.8.4. Email groups

You can manage a group of email recipients via the **Email Groups** page:

- Add a name and a description to create the group
- Add a list of email addresses

The email group is now available and can be selected where email groups are selected.

See for example *Global settings*, for:

- Webex App email to specify recipients of generated CSV files.
- Number Inventory Alerting - email group to receive alerts.

Related topics

- *Add a SMTP server*
- *Global settings*

2.9. Associate / disassociate devices to application users

High level admins can make available a utility to associate and disassociate devices to CUCM application users so that it displays on administrator menus on the GUI.

Important: The utility is available only at site level, so that only devices and device profiles at that level are available for management.

The utility simplifies the association and disassociation of a large number of Phones, CTI Route Points, or Device Profiles to app users (`device/cucm/AppUser`). This would for example be in the case of large contact center deployments.

The model to expose on the administrator menu layout is `view/AddRemoveDeviceAppUser`. Access Profile updates are needed for the default administrator access profiles.

1. Select an **Application User** to manage.

   The list of application users are those at and above the current site hierarchy.

   You can associate **Devices** and **Device Profiles** with the same application user by means of side-by-side transfer boxes.

2. Choose the **Action**: either to associate or disassociate devices or device profiles.
   - When associating, only devices or device profiles not associated to the user at the current hierarchy are shown in the **Available** transfer box.
   - When disassociating, only devices or device profiles associated to the user at the current hierarchy are shown in the **Available** transfer box.
3. Optionally, add a substring filter in the **Filter Devices** or **Filter Device Profiles**.

4. Manage the items in the **Available** and **Selected** transfer boxes.

   Note: The transfer boxes support up to 5000 initial results. More items may be available. In this case, specify a case-insensitive *contains* search to filter results, even if the list contains more than 5000 results.

5. Save your changes.

   Inspect the transaction log to verify the action. The **Detail** message includes the username of the app user.

Note: A database lock is used in the resulting transaction in order to ensure parallel transactions do not conflict.

Related topics

For details on customizing menu layouts, see *Create a Menu Layout* in the Core Feature Guide.

2.10. INI purge tool

Number Inventory entries can be deleted in bulk by using the INI Purge Tool. High level administrators with access to the view called `INI_Purge_Tool_View` can expose this tool for user by updating menu layouts and access profiles for administrators.

Important: This Tool has been designed to remove DNs *without* taking into account the Status of the DN, so that DNs can be completely removed and then redeployed. In normal circumstances you cannot delete a DN if it is in use by a Phone.

For example, DNs may have been deployed incorrectly at Site level instead of at Customer level. You can therefore “Purge” all the DNs entries at the Site and re-add them at Customer level. The DN Audit Tool then needs to be run to update the DNs to the correct Status.

Using the INI purge tool

1. Navigate to the required hierarchy and select the **INI_Purge_Tool_View**.
2. From the **Ini Purge Tool View** form, an **INI Option** can be selected.

   Note: Special characters in INI entry numbers are also supported, for example `1*6010`.

   The drop down list options are:

   - **All INI at Hierarchy**: applies to all INI numbers at the current hierarchy.
   - **Range of INI Entries**: exposes **Range Start Number** and **Range End Number** drop down lists to choose the range.

- **Picked List of INI Entries**: exposes **Picked INI Entries** transfer boxes: **Available** and **Selected**. to choose a selection of entries.
- **Specific INI Entry**: provides a **Specific INI Entry** drop down list to select a single entry.

3. Click **Save** to carry out the purge task.

Inspect the log from the **Transaction** menu - the **Action** carried out is a Bulk Delete from the Internal Number Inventory. Deleted numbers are removed and are not available from the **Internal Number Inventory** list view.

Note: Where more than 1000 entries are to be purged, the transaction will show sub-transactions deleting the entries in chunks of 1000. A successful sub-transaction then shows the message per chunk: “[1000 / 1000] were deleted successfully”.

---

3. System Monitoring

3.1. System monitoring configuration

3.1.1. Overview

A `sysadmin` administrator can access the **System Monitoring** menu (menu model: `data/SystemMonitoringConfig`) to manage:

- A number of alerts that will trigger SNMP traps
- Metrics collection

Default values are set in an instance called **Global**.

3.1.2. Alerts

From the **Alerts** tab on the **Configuration** menu, the following settings and options can be managed:

- **Database**:
  - **Database Size Threshold (GB)**: Size when alert is enabled. The default size is 200GB.
  - **Database Index Size Threshold (GB)**: Size when alert is enabled. The default size is 50GB
- **Transactions**:
  - **Transaction Queue Size Threshold**: Count when alert is enabled. The default count is 500.
  - **Maximum time in 'Queued' state (hours)**: Time since last transaction update of a queued transaction before an alert is enabled. The default time is 6 hours (maximum:48h, minimum: 1h).
  - **Maximum time in 'Processing' state (hours)**: Time since last transaction update of a processing transaction before an alert is enabled. The default time is 6 hours (maximum:48h, minimum: 1h).
  - **Transaction Failures to Alert**: errors for operations on model types.

    The default operation is all **Import** operations on all data models (`data/*`)

For the configured alerts, *platform-level* notifications can be generated upon failure of the transaction in two ways - using the **notify** command on the platform CLI:

- SNMP traps
- Email message

Note: Either or both notification types can be configured.

Refer to the *Warnings and Notifications* and *SNMP Configuration* topics in the Platform Guide for notification setup, details and examples of the SNMP traps.

Note: It may take up to 2 hours from the time that a transaction is considered hung/stuck to the time that an alert is created. Thereafter, there will be at least one alert created within every clock hour. The subsequent alerts will not necessarily be fixed 60-minute intervals.

3.1.3. Metrics collection

The **Metric Collection** tab shows:

- A configurable **RIS API data collector interval** which is the time interval that the real-time information (RIS) data collector service polls the Unified CM to obtain the latest phone registration status information for phone instances stored in in the VOSS Automate database.

  The value is in seconds and the default interval is: `43200` seconds (12 hours) Refer to the Best Practices Guide for further information if this interval needs to be modified.

  Note that the **Cisco RIS Data Collector** service needs to be enabled and running on the Unified CM publisher.

  The RIS data collector service updates the current registration status and/or IP address from the Unified CM Registration Status and IP address at the specified interval - for all clusters in the system. The **Phones** list view show **Registration Status** and **IP Address** columns containing this data.

  The status of the collector service can be checked with the the platform CLI **app status** command - seen as `voss-risapi_collector` in the example console output snippet below:

```
platform@VOSS:~$ app status
selfservice v19.3.2 (2020-04-18 19:27)
   |-node                  running
voss-deviceapi v19.3.2 (2020-04-18 19:30)
   |-voss-cnf_collector    running
   |-voss-queue            running
   |-voss-risapi_collector running
   |-voss-monitoring       running
   |-voss-wsgi             running


...
```

  Note:

  – There is an **Activate Phone Status Service** check box in `data/Settings` that is selected by default. Real time data collection is available when this check box is selected. Phone status data is then fetched directly from the Unified CM and shown on the **Phones** list view. See: *Activate phone status service*.

There is also a macro function `get_phone_status` available to return this Phone data, given as input parameters:

- a phone PKID
- followed by a comma and then exactly one RIS API field name.

  The fields below are for example used in the VOSS Automate Admin Portal list view of Phones:

  * `status`
  * `ip_address`
  * `cm_node`

  To see a full list of available fields, run the macro function without RIS API field names or refer to the Cisco RIS API documentation.

  For example:

```
{{fn.get_phone_status
      5ca2b90bce894e0014d488fb,
      status}}
```

  output: “Registered”

3.2. System monitoring database statistics

The `sysadmin` user has access to the **Database Statistics** menu (menu model: `data/MetricDatabaseCollectionStats`) that shows the *weekly and monthly* collection metrics internal database collections.

Note:

- Mongo database collections are similar to tables relational databases.
- Mongo database documents are similar to rows in relational database tables.

- The **Group Number** corresponds to the number of the **Grouping** - month or week of the year - respectively.
- The **Top Collections by Data Size** list shows individual data collections by name and sorted by size - displayed in bytes.
- The **Top Collections by Index Size** list shows individual data collection indexes by collection name and sorted by index size - displayed in bytes.
- The **Top Collections by Number of Documents** list shows data collection names by collection name and sorted by number of documents - displayed as an integer.

The **Configuration** menu allows for SNMP alerts to be configured if thresholds are reached in total Data Size and Index Size.

3.3. System monitoring model counts

The `sysadmin` user defines a list of **Models to Aggregate** on the **Metric Collection** tab of the **Configuration** menu (menu model: `data/SystemMonitoringConfig`). By default, the following models are included in the list:

```
device/cucm/CallPickupGroup
device/uccx/Agent
device/uccx/Team
device/uccx/ContactServiceQueue
device/uccx/ResourceGroup
data/BaseSiteDAT
device/cucm/User
device/cucm/Phone
device/cucm/HuntPilot
data/InternalNumberInventory
data/HcsDpE164InventoryDAT
device/cucm/Line
device/cucm/User
device/cucm/DeviceProfile
device/cuc/User
device/spark/User
```

For these models, counts are collected daily. Daily, weekly and monthly aggregates *at a particular hierarchy: Provider, Customer or Site* are then stored as a **Grouping**. In other words, the counts include all hierarchy nodes below the particular hierarchy.

In the instance details, the **Group Number** is then the nth day, week or month of the year.

Note: For daily aggregates, only the most recent daily aggregate of the week is stored. Previous daily aggregates are aggregated into this daily aggregate and are then deleted.

A particular daily or weekly Model Counts instance will then show the **Average Count** of instances as specified in the **Models to Aggregate** list.

The data is also displayed as usage charts on the Admin Portal.

3.4. Automate cluster status

The *sysadmin* user has access to the **VOSS Automate Cluster** page (model: `data/MonitoringCluster`), which displays aggregated Automate cluster Round Trip Delay Time (RTT) metrics at hourly and daily intervals.

Note:

- The retention period for data is 1 month.
- This metric uses the same data as the `cluster check` CLI command. See *Cluster Check* in the Platform Guide.

- The **Interval** indicates the time interval for the entry: hourly or daily.

  Hourly intervals aggregate samples collected 15 minute intervals. Daily intervals aggregate hourly intervals.

- The **Start** and **End** values are the time stamps for the interval. The difference corresponds with the selected **Interval**.
- The **Source host** and **Destination port** are the IP addresses of the nodes in the cluster.
  - Port 27020: database connectivity
  - Port 8443: HTTPS connectivity

  For example, if **Source host** is `192.168.100.7` and **Destination port** is `192.168.100.6:27020`, then database connectivity is monitored between `192.168.100.7` and `192.168.100.6:27020`.
- The **Failures** is the number of connection failures between **Source host** and **Destination port** over the interval.

  Failures would also generate alerts if an SNMP trap is configured.
- The **Avg TCP RTT (ms)** is the average RTT in milliseconds over the interval for a successful connection.

3.5. UC apps reachability

Provider administrators and higher have access to the **System Monitoring** menus:

- **Unity Connection** (`data/MonitoringCuc`)
- **Call Manager** (`data/MonitoringCucm`)
- **LDAP** (`data/MonitoringLdap`)

The displayed data is the aggregated reachability and Round Trip Delay Time (RTT) for a UC app. Reachability is tested from *all* unified nodes.

- The **Interval** indicates the configured time interval for the entry: hourly or daily.

  Hourly intervals aggregate samples collected 15 minute intervals. Daily intervals aggregate hourly intervals.

- The **Start** and **End** values are the time stamps for the interval. The difference corresponds with the selected **Interval**.
- The **Unified Node** is the node name of the source unified node.
- The **Host name or IP** is the IP addresses and port of the UC app.
  - Port 8443: HTTPS connectivity
- The **Avg TCP RTT (ms)** is the average RTT in milliseconds over the interval for a successful connection.

  The details of a data instance also show the maximum RTT (**Max Tcp RTT**) during the interval.

  If the average RTT exceeds 400 ms, an alert is also triggered.

- The **Failures** is the number of connection failures to the UC app **Host name or IP** over the interval.

  For example, if the number of failures is 4 and the **Interval** is set to *hourly*, then all 4 reachability checks for the interval failed. Equally, if the **interval** is set to *daily* and all reachability checks failed, the value is 96 (24 x 4 hourly).

  The details of a data instance with failures show the network error messages in the **Errors** group.

- The **Located At** column shows the hierarchy of the UC app.

Failures would also generate alerts if an SNMP trap is configured.

3.6. Worker queue

Provider administrators and higher have access to the **Worker Queue** menu (`data/MonitoringQueue`).

Data is retained for 1 month.

The displayed data is the aggregated number of transaction requests per processing node and configured interval.

- The **Interval** indicates the configured time interval for the entry: hourly or daily.

  Hourly intervals aggregate samples collected 15 minute intervals. Daily intervals aggregate hourly intervals.

- The **Start** and **End** values are the time stamps for the interval. The difference corresponds with the selected **Interval**.

  For the same **Start** and **End** timestamp, an entry is shown for each **Processing node** showing the combination of **Status**, **Priority** and **Tx level**.

- **Status**: Transaction process status: “Queued” or “Processing”.
- **Priority**: Transaction priority: Low, Normal or High
- **Tx level**: Transaction level: either parent or child transaction.
- **Processing node**: the node that carried out the transaction.
- **Avg Transactions**: the average of the **Max Transactions** and **Min Transactions** for the selected **Interval**.

  Note: The record detail also shows the **Min Transactions** and **Max Transactions** for the configured interval. These values reflect the sampling in the interval. For example, if the **Interval** is hourly and **Min Transactions** is 0, then one or more of the 15-minute sampling intervals recorded no transactions.

- The **Located At** column shows the hierarchy at which the transaction was run.

  Note: If the value shows as `sys(System)`, this includes all transactions at `sys` level and lower.

Also refer to the use of the **voss worker** and **voss workers** commands in the Platform Guide.

3.7. Login sessions

Provider administrators and higher have access to the **Login Sessions** menu (`data/MonitoringSessions`).

- The **Interval** indicates the configured time interval for the entry: hourly or daily.

  Hourly intervals aggregate samples collected at 15 minute intervals. Daily intervals aggregate hourly intervals.

- The **Start** and **End** values are the time stamps for the interval.
- **Interface**: The user interface to which the limit applies: `administration` or `selfservice`.
- **Max sessions**: maximum number of sessions, for example in any of the 15-minute sampling records during an hourly interval.
- **Utilization %**: utilization percentage for the interval in accordance with the defined *customer* session limits (max sessions / defined limit).

  By default, the following limits apply:

  – per customer administration : 10

  – per customer selfservice : 1000

- The **Located At** column shows the hierarchy under which all sessions for a given entry are being counted. The sessions being counted are either by administrator or Self-service.

Note: The values reflected for at the `sys` hierarchy are global totals across all hierarchies, and the utilization percentage is calculated against the global limits.

Also refer to the use of the **voss session-limits** commands, for example to show and manage session limits, in the Platform Guide.

4. Customization Overview

4.1. Feature package customization

4.1.1. Overview

This topic describes the components available for customization in Automate.

Related topics

- Field Display Policies in the Core Feature Guide.
- Configuration Templates in the Core Feature Guide.
- Device Models in the Core Feature Guide.
- Menu Layouts in the Core Feature Guide.
- Dashboards in the Core Feature Guide.
- Introduction to role-based access in the Core Feature Guide.

Models

Automate makes use of components called *models* that can be added and modified.

The table describes the primary types of models:

Model Type	Description
Data Models	Used to capture and store data.
Device Models	These models represent components of available network devices.

Field display policies and configuration templates

Higher level admins can use the following mechanisms for customization:

Mechanism	Description
Field Display Policies	Admins can use field display policies (FDPs) to define how different attributes of a form are displayed: • Visibility (hidden / visible / read-only) of attributes • Field names • Related help text • Ordering, grouping, and layout FDPs are applied to different roles using menu layouts and dashboards.
Configuration Templates	Admins can use configuration templates (CFTs) to define how values for attributes are obtained: • Fixed or default values • Derived from data in the system; optionally combined with input from users or Device Model events using pre-defined macros. See *Macros* CFTs are applied to different roles using menu layouts, dashboards, and pre-defined workflows.

4.1.2. Customization task overview

A high level administrator can follow these high level steps for customization:

1. Decide how an existing user input form should be customized.

   In this case, inspect the input form to identify field name, visibility, order, and default values.

2. Identify the user/s requiring a modified input form.
3. Identify the user role and menu layout associated with the relevant user/s.
4. If the menu item to be customized shows a field display policy (FDP) and configuration template (CFT) associated with the item, then clone these to start their customization.

   Note: There is a unique constraint on the name of the clone per hierarchy level, so the same name as the original can be used on another hierarchy, but a new name is needed at the same hierarchy.

5. Create, or modify the cloned FDP and CFT according to identified requirements to create new instances of these.

   Note: Note uniqueness constrains per hierarchy on the names of the clones.

6. Associate the newly created FDP and CFT to the menu item in the user's menu layout as specified in the user role. The menu layout may be created as a new menu layout that is only available at a specific hierarchy.

4.2. GUI Customization

4.2.1. Field display policies

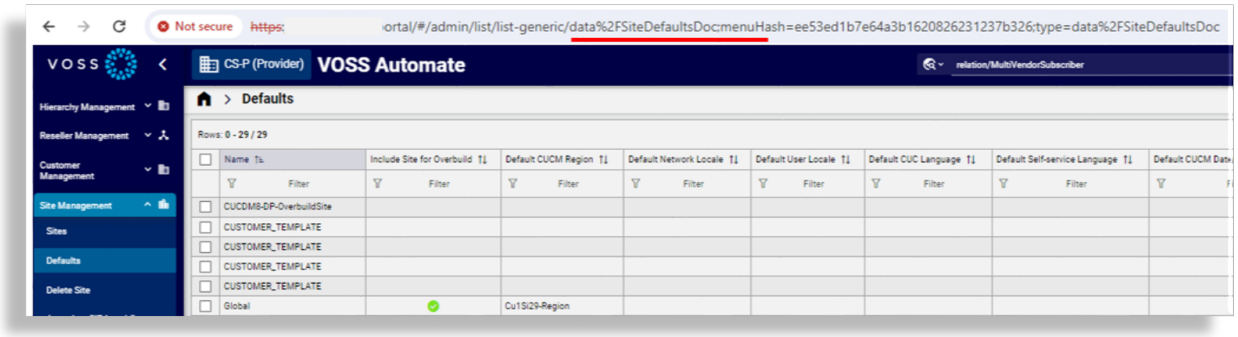
Tip: *Use the Action search to navigate Automate*

Overview

Field display policies (FDPs) define the layout and composition of fields, fieldsets, and groups on forms in the Automate Admin Portal GUI. To change these elements on the form, you can edit the FDP associated with the form, if allowed.

Note: FDPs are assigned to the model associated with a form or page in the GUI. FDPs can be added to Data models, Relations, and Views. For example, *data/SiteDefaultsDoc* is the model associated with the summary list view page and with the edit form for the site defaults.

If you want to change the layout of the form, you can edit the FDP for this model, if allowed. The layout of forms associated with some models cannot be modified via their FDP, for example, for `relation/LineRelation`, `relation/SubscriberPhone`, `tool/Transaction`, or `view/BulkAddUser`.



Via the FDP, you can define whether to group or disable fields, add online help text to a field, add a label to a field, or move a field up or down on the form.

You can apply one or more FDPs to a particular item type to present different views of the same form. You can apply a FDP for a menu layout and role so that users with this role are presented with a view of the form (defined by the FDP) for their user role when they log in. For example, a system may have users at Provider, Customer, and Site administration hierarchy levels - all of whom may access the same items, but perhaps some item fields need to be hidden for admin users at some levels. Therefore, you can create and apply a specific FDP to a menu layout designed for admin users at these levels.

You can clone an existing FDP to quickly create one, then modify the clone as required, and choose this new FDP for the model on a user's menu layout. In this way, the user's view of the GUI can be modified for their level of access to the model, from the menu.

Note: The list view column header will also show the field title from the FDP if the field belongs to the list of summary attributes.

Field display policy naming convention

The name of a FDP must be unique at each hierarchy. You can however have FDPs with the same name at different hierarchies.

FDPs that have the name `default` will apply to their associated model, by default.

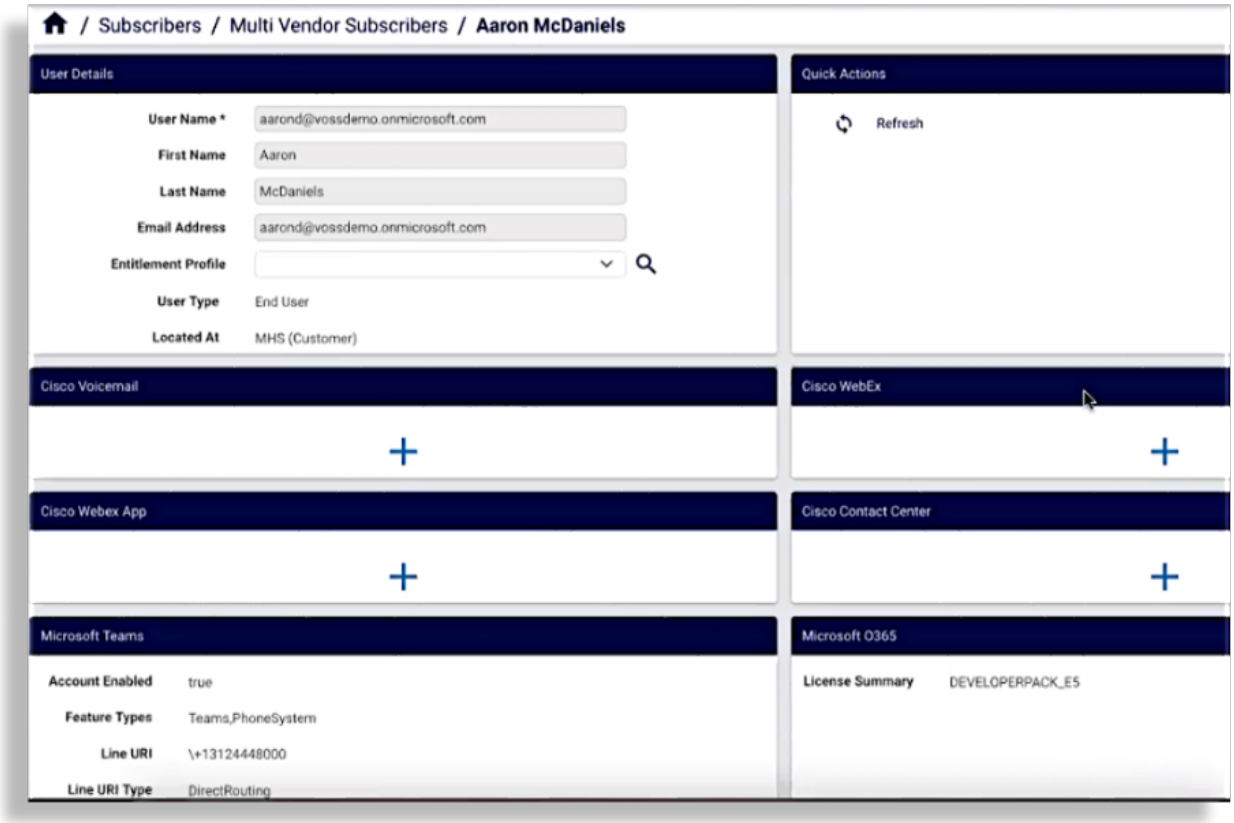
Display groups as tabs or panels

The **Field Display Policy** configuration screen provides a **Display As Groups** setting that allows you to define the default layout of some forms, as either tabs or panels.

Tabs	Each group of fields and/or fieldsets displays on a tab. The group title is the tab title.
Panels	Each group of fields and/or fieldsets displays as a panel in a 2-column list of panels on the GUI. The group name is the panel header.

Note: If no option is selected, the default is **Panels**, except for some models, which display groups as tabs by default. On some forms, depending on your user type and the model where the FDP is applied, you can click a toolbar icon in the GUI (or select from the action overflow menu) to switch between a panel or tab layout. The layout you choose is preserved when you log out and log in again.

The image shows an example of a form where groups display as *Panels*:



The Automate system default is *Panels*, except for the following models, which have as their default setting, *Tabs*:

- `view/GlobalSettings`
- `data/SiteDefaultsDoc`
- `data/ucprep_UC_Profiles`
- `relation/DP_REL`
- `data/HcsDpDialPlanSchemaDAT`
- `data/HcsDpDialPlanSchemaGroupDAT`
- `tool/BulkLoad`
- `tool/DataImport`

The table lists models where the layout of elements of add and/or edit form types can't be modified via FDP. These form types also do not have the action element available on the GUI to switch the form view between tabs and panels.

Model Name	Form Type	Default
`relation/LineRelation`	edit	Panels
`relation/MultiVendorSubscriber`	edit	Panels
`relation/PexipConference`	add, edit	Panels
`relation/SubscriberDeviceProfile`	add, edit	Panels
`relation/SubscriberPhone`	edit	Panels
`tool/Macro`	add	
`tool/Transaction`	edit	
`view/AddSubscriberFromProfile`	add	
`view/BulkAddUser`	add	
`view/HcsVersionVIEW`	add	
`view/MenuDiff`	add	

Related topics

- Field display policy settings in the Advanced Configuration Guide
- Introduction to the Admin Portal User Interface in the Core Feature Guide

Add and edit field display policies

This procedure adds and edits a field display policy (FDP).

1. Log in as Provider administrator or higher.
2. Choose the relevant hierarchy.
3. Go to **Field Display Policies** to view the list of existing FDPs.
4. **Do you want to . . .**

- **Edit an existing FDP**? Click on the relevant FDP in the list view to open the configuration form. Update the FDP as required, then save your changes.
- **Add a new FDP**? Go to step 5.

5. To add a new FDP, from the list view, click the toolbar Plus icon (+) to open the configuration form for the new FDP you're adding.
6. Configure the FDP:
   a. Fill out a name for the new FDP, and optionally, a description.
      - If the FDP name is `default`, it is applied by default to the target model type you choose on the form.
      - FDPs at the same hierarchy must have a unique name. FDPs at different hierarchies can share the same name.
   b. Choose the target model type to associate with the FDP.

      Note: The target model type defines the fields available for use in the FDP.

   c. At **Display Groups As**, choose how groups should display on forms. Options are Tabs or Panels.

      Note: The default is **Panels**, except for a selection of models, where the default is **Tabs**. Within a tab or a panel, you can add a combination of fields and/or fieldsets (within one or more groups).

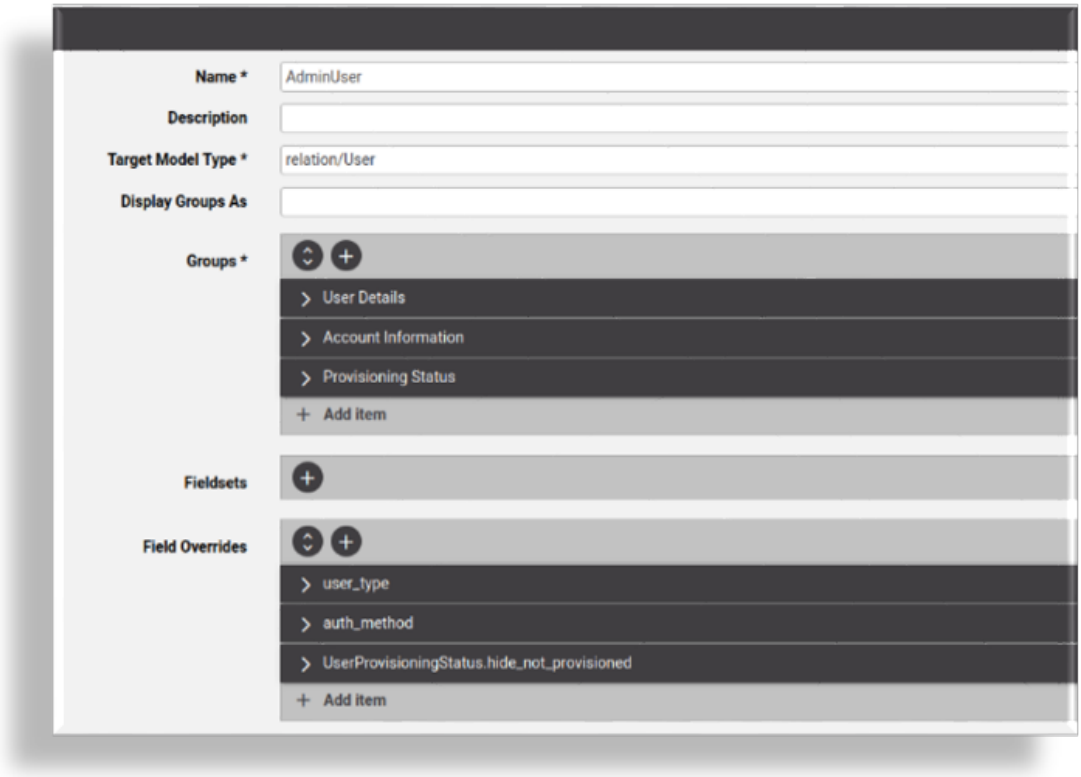
   d. Add groups, one or more, required:

      At **Groups**, click the Plus icon (+), then configure the group.

      Note: Groups that describe a collection of attributes display together on the GUI. All fields in the FDP must belong to a group.

      At the form level, groups all display either in panels or tabs, depending on the option selected in **Display Groups As**, and provided the model allows the option you choose.

      You can copy or re-order (move up or down) groups, fields, and fieldsets.



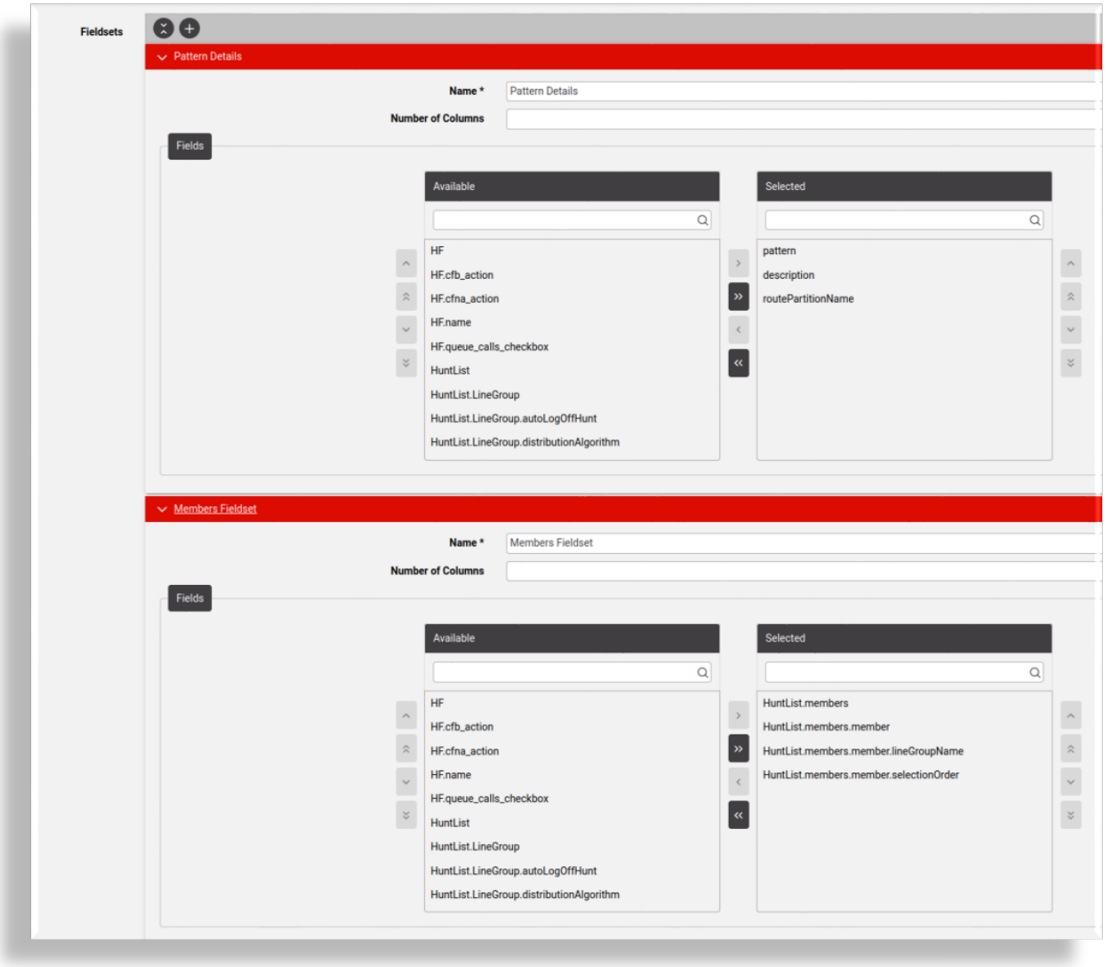
The table describes the group configuration options:

Component	Description
Title	Mandatory. Fill out label text to display for the attribute on the new tab. If a group displays as a tab in the Admin Portal, the value defined for **Title** displays as the title of the tab.
Number of Columns	Fill out the number of columns for fields. The default is *1* (a single column). Fields in the **Selected** transfer box display in these columns.
Fields	Choose fields to add. Select then move fields from the **Available** field to the **Selected** field. The selected target model type defines the available fields. Use the **Move Up**/**Move Down** buttons to adjust the position of any field.

e. Add fieldsets, if required. Click the Plus icon (+) at **Fieldsets**, then configure the fieldset.

Note: Fieldset options (fields in the Available transfer box) show all field choices for the selected target model type. Fieldset names are added as choices within any group added to the FDP, and the fields display as a group in fieldsets in the panels.

The **Name** field is the name of the fieldset. First create fieldset, then add it to a group.



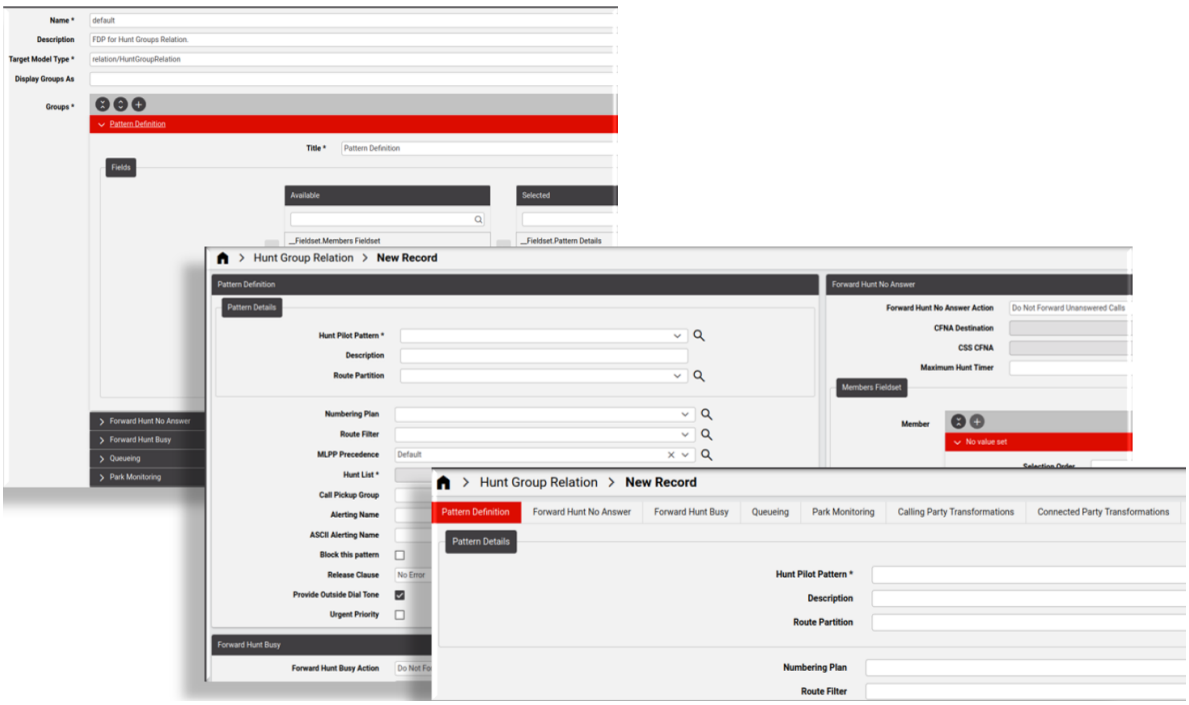
f. Add field overrides, if required. Click the Plus icon (+) at **Field Overrides**, then configure the field override.

The table describes configuration options for field overrides:

Component	Description
Field	Name of the model field to override. Options include those added to the **Selected** field for groups.
Title	New title to display for the field. If the FDP is called `default` at a hierarchy, the list view column header also displays this title if the field belongs to the list of summary attributes.
Help Text	New help text to display for the field. Leave blank to use the model attribute description.
Disabled	Sets the field as *read-only*.
Input Type	Overrides the input type of the field. Select an option to choose how the input field displays, for example, radio button, grid, multi select.

7. Click **Save**.

Once saved, the FDP can be applied by selecting it in a menu layout available to a role.



Clone a field display policy

This procedure creates a copy, or clone, of an existing field display policy (FDP) to create a new FDP, starting with the configuration of the FDP you're cloning.

1. Login as Provider administrator or higher.
2. Choose the hierarchy.
3. Go to **Field Display Policies** to view a summary list of existing field display policies (FDPs).
4. Click on the FDP you want to clone.
5. Choose **Actions > Clone**.
6. Update the necessary fields for the cloned FDP.
7. Click **Save**.

   You can apply the cloned FDP by choosing it in a menu layout available to a role.

Rules for creating field display policies

When creating groups and selecting the field transfer boxes of a group, a number of rules apply.

Note: Regarding notation, if the fields belong to objects or arrays, the names in the transfer boxes are shown in dot notation. Refer to the target model type on-line help field reference to distinguish object types from array types.

To understand the rules, consider a selected Target Model Type with the fields as listed below. Where the name starts with "A", the field is an array and where it starts with an "O" it is an object. The values "x", "y", "z" are also objects. The field "F" is neither object or array.

- A, A.x, A.x.b, A.x.c, A.x.d, A.y.r, A.y.s, A.y.t
- F
- O, O.v, O.z, O.z.a, O.z.b, O.w.d

Inclusion rules

The following inclusion rules apply:

- If a parent object or array field is included, the parent and all its children will be displayed in the GUI.

  Example: if O.z is selected, O.z is saved as the fields and the GUI will display O.z and also inner fields O.z.a and O.z.b.

- If a specific selection and order of child elements are required, select these child elements and order them.

  Example: if O.w.d, O.z.b, F are selected, these three fields are saved in that order in the FDP group fields and the GUI shows only the inner field O.w.d, followed by the inner field O.z.b and lastly the field F.

- Inclusion of child fields in a group without the inclusion of the parent fields will display these child fields at the root level of the form.

  Example: if O.w.d, O.z.b are selected, these fields are saved as is in the FDP group fields list and only the inner fields O.w.d and O.w.b are shown in the GUI.

- Array children fields without their parent fields will be ignored by the GUI. Therefore, if the child fields of an array field are selected, the parent field should also be selected.

  Example: if A.y.s, A.y.t are selected, A and A.y should be selected.

- Array fields may not be split into different groups.
- The parents of fields cannot be in one group and its children in another.

  Example: O.z cannot be in Group 1 if O.z.a, O.z.b and O.w.d are in Group 2.

- Fields of the same object and members of the same array type cannot belong to more than one group.

  Example:

  - If A.y.s is selected for Group 1, then A.y.t cannot be selected for Group 2.
  - If O.z.a is selected for Group 1, then O.z.b cannot be selected for Group 2

- You can split the first level children of object fields into different groups.

  Example:

  - O.v can be in Group 1 while O.z is in Group 2.
  - For second level children: O.z.a can be in Group 1 and O.w.d can be in Group 2.

- To hide a field do not move it to a Selected box.

  Example: To hide O.z.b, select O.z.a, O.w.d.

Ordering fields in a field display policy

You can move fields or fieldsets in a group (up or down on the form) by clicking the **Move Up** / **Move Down** buttons at the group level or in the transfer boxes.

Ordering child and parent fields depends on the presence of siblings, other parents, and children. If a child is selected in a group and not its parent, but a sibling of that parent is selected, then the sibling's order will affect the order of the fields.

The logic of order resolution starts from parents to children, according to the rules below.

For example, we select fields in this order in Group 1:

C.z, A.x.b, A.x.c, B, A.y, A.x, C, C.w

Result:

- Parent fields on their own are considered first, hence our initial order is B, C.
- However, parent A is not selected; only the children. We determine where A was mentioned. In this case the children of parent field A were mentioned before the parent fields B or C. Hence children of A will eventually be ordered before B and C.
- Next we consider the selected first level child fields: C.z, A.y, A.x, C.w. The order becomes: A.y, A.x, B, C, C.z, C.w
- We now move down the levels: A.x.b, A.x.c.

Thus the final display order will be:

A.y, A.x, A.x.b, A.x.c, B, C.z, C.w

Further examples below illustrate the presence of parents, siblings and children on the selected order.

- We add fields C.w, A, C, B, A.x, A.y.

  Result: The order is: A, A.x, A.y, C, C.w, B.

- We add fields A.x.b, A.x.c, A.y, A, B

  Result: The order is: A, A.x, A.x.b, A.x.c, A.y, B.

Note: Note that A.x was added and that A.y is placed after A.x, since the children were ordered before A.y while A.x was never selected.

4.2.2. Field display policy settings

Input Type	Can apply to Data Type
Default	
Multiselect	Array
Radio button	String - choices
Sequence	Array
Transferbox	Array

Sequence is a list of drop-down boxes created to allow for the selection of an item in the array: one is selected from each displayed drop-down list.

Transferbox is a side-by-side **Available** and **Selected** list boxes with controls to select and unselect items.

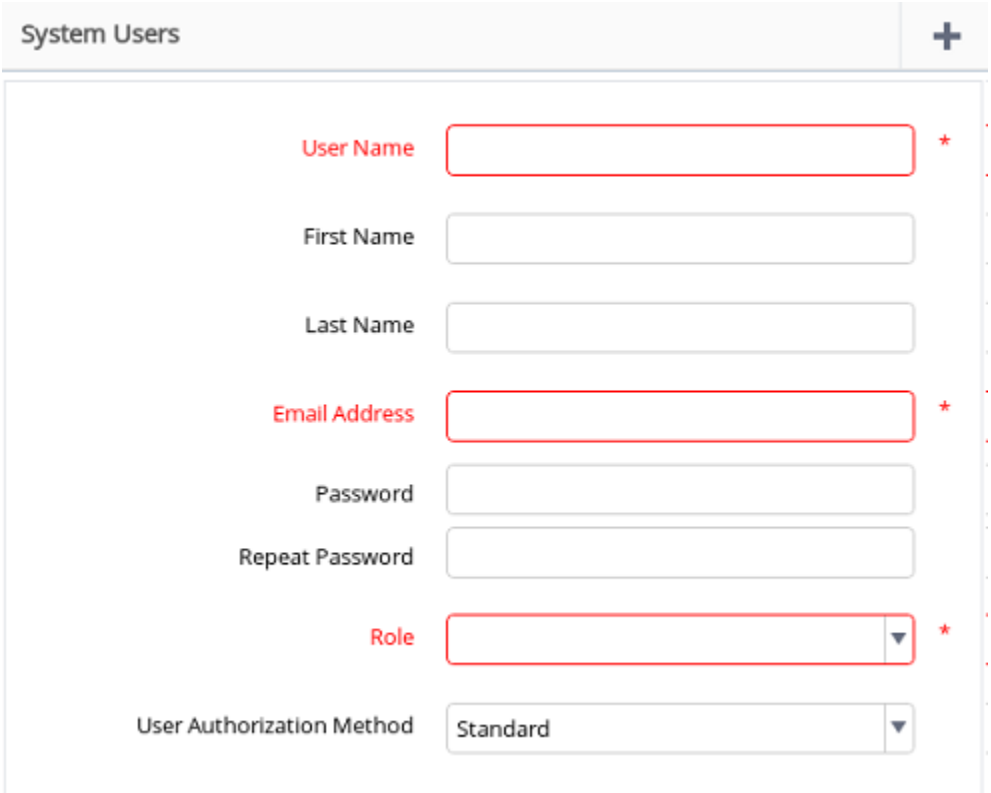
4.2.3. Example field display policy clone

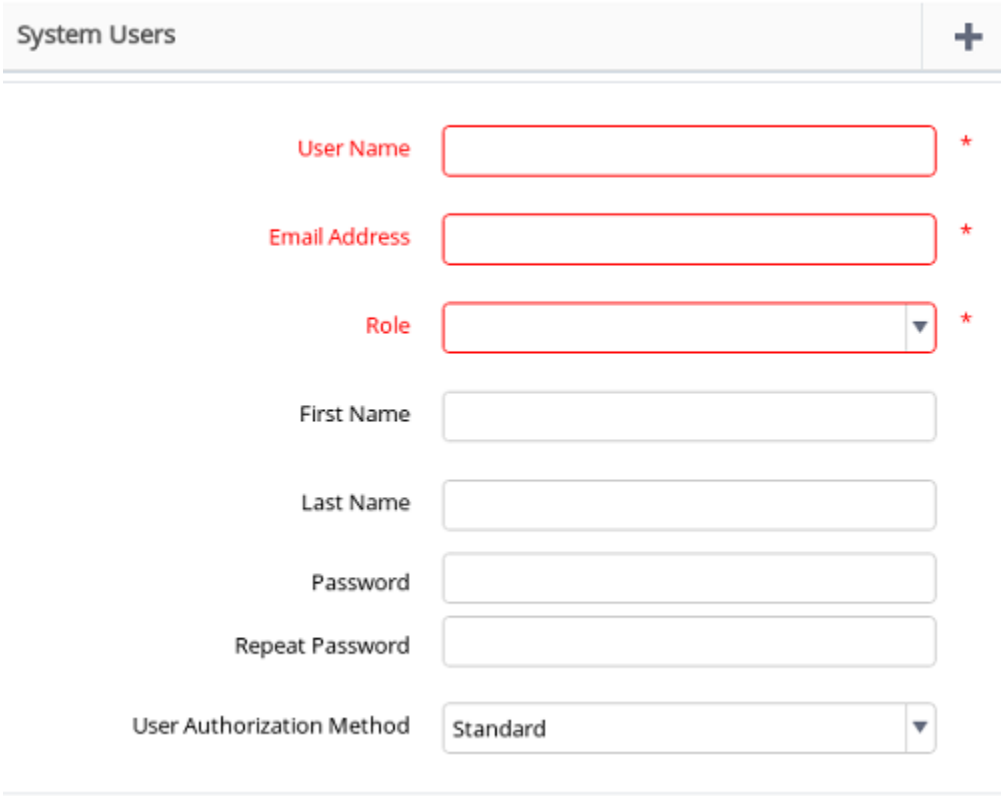
The following example shows the **System User** interface before and after a cloned Field Display Policy has been modified and applied to an item on a Menu Layout change.

The table below shows the changed original and new order on the design form of the Field Display Policy so that all the mandatory fields display at the top of the item form.

Field Name	Orig. Order	New Order
User Name	1	1
First Name	2	4
Last Name	3	5
Email Address	4	2
Password	5	6
Repeat Password	6	7
Role	7	3
User Authorization Method	8	8

The screenshots show the example original and changed forms.





4.2.4. Configuration templates

Tip: *Use the Action search to navigate Automate*

Introduction to configuration templates

Configuration templates (CFTs) are used to define values for the attributes of any model.

Values can be fixed values, or existing macros visible from the hierarchy (for example, customer or site), where the CFT is applied.

CFTs allow you to define default values for items exposed in the Admin Portal (visible, hidden, or read-only). CFTs provide a way to map data from data input via the Admin Portal or device model events to other models or provisioning workflows in the system.

You may want to hide the attributes of a model while setting them to a specific fixed value (for example a hard-coded setting); or you may wish to derive the value based on a macro (for example, look up the value based on data in the system).

Examples

- A model with an attribute defined as a date string; a CFT for the attribute can be defined as a macro `{{fn.now \"%Y-%m-%d\"}}` in order to set the current date stamp as the value, such as `2013-04-18`. Designers can access reference material for details on macros.
- A model such as Quick Add User, which limits user input to a few fields while deriving the value of other hidden attributes from various CFTs that are each applied to different underlying models that make up a user, such as voicemail account settings, conference account settings, phone, line, or device profile settings.

When adding or updating an instance of the model, the CFT enabled on the model is applied.

For array elements of data models, a list and a variable can be specified to be looped through so that a value is applied to each element in the model array.

You can create one or more CFTs for a model, and these can be used as needed. CFTs can also be applied to models in the design of, for example, provisioning workflows.

A menu layout that can be associated with a user role can also apply a CFT to a model that is selected as a menu item.

Provider administrators (and higher level admins) can quickly add a new CFT by opening a similar CFT (via, say, the Configuration Templates menu), then making a copy (clone) of it, and customizing the clone to create a new CFT.

Administrators at levels above the site admin can also customize these templates, including Field Display Policies (FDPs).

Note:

- When modifying CFTs in the Admin Portal, numerical values must be filled out using the `fn.as_int` function, for example:

  ```
  {{ fn.as_int 14 }}
  ```

- In a multi-cluster environment, CFTs that result in device model drop-down lists in the Admin Portal may contain duplicates. Any duplicated item can be selected by the user.

Related Topics

- Cisco Quick User in the Core Feature Guide
- Quick Add Groups in the Core Feature Guide

Add a configuration template

This procedure clones and edits an existing configuration template (CFT) to create a new CFT.

1. Log in to the Admin Portal as Provider administrator or higher.
2. Go to **Configuration Templates** to view the list of existing CFTs.
3. Click on a configuration template (CFT) you wish to clone, and view its details.
4. Click **Action > Clone**.

5. Edit the required generic fields, such as **Name**, **Description**, **Target Model Type**, and the fields specific to the selected model type. See *Configuration template settings*

   Note: Some fields are populated based on specific conditions. For example, when creating a device instance CFT in a multi device or clustered environment, drop-down values in the CFT that originate from a device will be the values from *all* the devices in the cluster. For this reason, the list may include duplicates; in this case, you can choose any duplicate, if required.

6. Click **Save**.

   The new, cloned CFT appears at the selected hierarchy level.

Example: Add a CFT for a Cisco 6941 SCCP phone

1. In the Admin Portal, go to the relevant hierarchy, the hierarchy where the Cisco UCM you want to use exists.

   Note: This step is required if the fields are to populate values because some of the values are derived from the actual device model through the API.

2. Click the **Default CUCM Phone Template**, and then click **Action > Clone**.

   Note: Don't save your changes yet.

3. Change the template **Name** and **Description**.
4. Edit the template fields:
   - From the **Device Protocol** drop-down, choose **SCCP**.
   - From the **BAT Phone Template** drop-down, choose **Standard 6941 SCCP**.
   - From the **Device Security Profile** drop-down, choose **Cisco 6941 - Standard SCCP Non-Secure Profile**.
   - From the **Product** drop-down, choose **Cisco 6941**.
   - From the **BLF Presence Group** drop-down, choose **Standard Presence Group**.
   - In the remaining fields, use the cloned default values.

   Tip: You can type in the values if you know them; else, choose values from the list.

5. Click **Save**.

Configuration template settings

The table describes general fields on the **Configuration Template** page:

Note: Fields specific to the CFT for the selected target model type are excluded.

Title	Field Name	Description
Name *	name	The name that is given to the Configuration Template.
Description	description	A description for the Configuration Template.
Foreach Elements	foreach.[n]	Iterates over the list returned by the macro and appends array elements to the specified field.
Property *	property	The field property to iterate over.
Macro List *	macro_list	The macro that produces the list to iterate over.
Context Variable *	context_var	The context variable that will contain the data from the iteration.
Schema Defaults	schema_defaults.[n]	Applicable only when the configuration template is used directly in API requests. This attribute contains a list of paths to the properties of the template section that must be used to enrich the default values of the schema. All paths specified must refer to array attributes.
Target Model Type *	target_model_type	The target model type and name that the Configuration Template applies to.
Merge Strategy	merge_strategy	Determines how this CFT will be merged into another CFT when it is being processed in a PWF. Default: additive.
Template *	template	The contents of the template, such as defaults and macros. The names shown in the template are determined by the attribute names of the Target Model Type.

4.2.5. Add a configuration template

1. Choose the hierarchy level at which you want to create a Configuration Template.
2. From the **Configuration Templates** list view, clone the required template.
3. Enter the **Name** (mandatory field) and **Description**, and verify the **Target Model Type**.
4. Use the **Custom feature usage identifier** array list input to add one or more values that can then for example be used in GUI Rules macros to filter choices in drop lists, for example by `QAS` and `SAP` below:

```
{# data.ConfigurationTemplate.name |
   target_model_type:device/cucm/User,feature_usage:QAS #}
```

(continues on next page)

(continued from previous page)

```
{# data.ConfigurationTemplate.name |
   target_model_type:device/cucm/User,feature_usage:SAP #}
```

Note: Only add or modify the feature usage details for *custom features* which you add to the system. Changing the feature usage on Configuration Templates which are shipped with the product may have unintended behaviour changes for the standard set of features included with the product.

5. If the Configuration Template will be used along with other Configuration Templates that apply to the same **Target Model Type** in a Provisioning Workflow, select the **Merge Strategy** to be applied when these Configuration Templates are merged in the workflow:
   - **Additive** - if the Data Type of the Target Model is an array, the Configuration Template will add an array item to any existing array.
   - **Replace** - if the Data Type of the Target Model is an array, the Configuration Template will replace the array item of any existing array.
6. If required for model attributes that are arrays, add **Foreach Elements** by clicking **+** (Add) for each:
   a. Enter a **Property** that is the model array attribute to which the entry applies.
   b. Enter a **Macro List** to loop over. A macro list opens with {# and closes with #}.
   c. Enter a **Context Variable** name to store the current loop value of the macro list.

   The **Context Variable** is referenced as a macro with the *cft* prefix in the Configuration Template value for the array attribute, so that for each instance in the list loop, an array entry is created when the template is applied. For example: `{{cft.MyContextVar.name}}` Configuration Templates that are part of Provisioning Workflows can reference spreadsheet column values using the `input` syntax: `{{input.[entity attribute]}}`, for example `{{input.username}}`.
7. In the **Target Model Type** section, enter (default) values for those displayed attributes that these should be applied to when the template is used in conjunction with the target model. If the user has access to macros, the values can be macros to determine values.
8. Click **Save** on the button bar to create and save the Configuration Template.

The created Configuration Template is available to be applied to the model by for example using it in a Menu Layout or Provisioning Workflow.

4.2.6. Configuration templates for array updates

When creating configuration templates for any of the models in the **Model Type** column below, it is not necessary to also set changed and existing values for *every* array item in the **Array Field** column.

Only values for the *changed* items are required. The attribute in the **Identifying Attribute** column will be used to maintain the existing array item values.

Model Type	Array Field	Identifying Attribute
RemoteDestinationProfile	lines.line	dirn
RemoteDestinationProfile	lines.line.associatedEndusers.enduser	userId
CtiRoutePoint	lines.line	dirn
CtiRoutePoint	lines.line.associatedEndusers.enduser	userId
Phone	lines.line	dirn
Phone	lines.line.associatedEndusers.enduser	userId
GatewaySccpEndpoints	endpoint.lines.line	dirn
GatewaySccpEndpoints	endpoint.lines.line.associatedEndusers.enduser	userId
GatewayEndpointAnalogAccess	endpoint.lines.line	dirn
GatewayEndpointAnalogAccess	endpoint.lines.line.associatedEndusers.enduser	userId
DeviceProfile	lines.line	dirn
DeviceProfile	lines.line.associatedEndusers.enduser	userId

4.2.7. Events

Events execute Provisioning Workflows in response to a trigger. The trigger is an operation that is carried out on a selected model. The available actions for a selected model define the list of available operations.

Event instances to trigger are searched for as follows: the lower of:

- the transaction hierarchy (in other words the transaction “breadcrumb”)
- the triggering resource hierarchy

When an Event instance is defined, the following values are specified:

- Workflow - the created Provisioning Workflow is selected.
- Active - if the Event is to be enabled, it is set to be active. This means events can be created without being active.
- Model Type - This is a part of the transaction. Operations on Model Types define the transaction.
- Operation - the available operations depend on the selected Model Type.
- Phase - the event can take place before or after the defined transaction. For example, if the Create Operation is available for the Model Type and the Phase is Pre Execution, then the event workflow is run *before* the Create Operation.
- Synchronous - if enabled, the transaction is only carried out if the HTTP response from the HTTP request in the workflow shows the request was received, in other words the response is of the format `HTTP 2xx`. If the Event is not set to be Synchronous, in other words it is asynchronous, the transaction is carried out regardless of the HTTP response.

> **Warning:** Always enable events as Synchronous, unless there is a good reason to set an event as asynchronous. Failed asynchronous events do not roll back the triggering transaction and can lead to unexpected issues.

A hierarchy context of the selected model is available to a Workflow in the case of events. In the case where a model allows for a Move operation and a Post Execution Phase is specified, this hierarchy value would be changed to the new hierarchy.

Consider for example the following transaction context data:

```
Pre event hierarchy: 5404f304c8e69d774458660e
Post event hierarchy: 5404f711c8e69d774458a92a

device/cucm/User Pre Asynchronous move:


Step 0 - Executing workflow (testEVPWF) with the following context data:
 {
    "input": {
        "device_pkid": "5404f345c8e69d7744586676"
    },
    "trigger": {
        "model_type": "device/cucm/User",
        "phase": "preexecution",
        "operation": "move",
        "trigger_model_type": "data/Event",
        "name": "testEV_cucmUser_Pre_async_move"
    },
    "resource_meta": {
        "model_type": "device/cucm/User",
        "pkid": "5405ba26c8e69d77445b8769",
        "hierarchy": "5404f304c8e69d774458660e",
        "device_pkid": "5404f345c8e69d7744586676"
    },


device/cucm/User Post Asynchronous move:


Step 0 - Executing workflow (testEVPWF) with the following context data:
 {
    "input": {
        "device_pkid": "5404f345c8e69d7744586676"
    },
    "trigger": {
        "model_type": "device/cucm/User",
        "phase": "postexecution",
        "operation": "move",
        "trigger_model_type": "data/Event",
        "name": "testEV_cucmUser_Post_async_move"
    },
    "resource_meta": {
        "model_type": "device/cucm/User",
        "pkid": "5405ba26c8e69d77445b8769",
        "hierarchy": "5404f711c8e69d774458a92a",
        "device_pkid": "5404f345c8e69d7744586676"
    },
```

The Provisioning Workflow is triggered from the lower of: the transaction hierarchy (eg. breadcrumb) and the hierarchy where the event instance is located.

To override this hierarchy in order that the workflow is run in the target hierarchy, use the `{{ resource_meta.hierarchy }}` macro for the workflow context hierarchy.

For example:

```
{
   "meta": {},
   "resources": [
      {
         "meta": {
             "model_type": "data/ProvisioningWorkflow",
             "pkid": "53fde2d02afa4356c87e45c4",
             "schema_version": "0.3.9",
             "hierarchy": "sys",
             "tags": []
         },
          "data": {
             "name": "__post_event_test",
             "parameters": {
                "max_workers": "1",
                "parallel": "false"
             },
             "workflow": [
                {
                  "entity_type": "model",
                  "hierarchy": "{{ resource_meta.hierarchy }}",
                  "entity": "data/Countries",
                  "method": "refresh",
                  "advanced_find_options": [
                      {
                          "model_attribute": "iso_country_code",
                          "mapped_value": "USA"
                      }
                    ],
                  "advanced_find_search_direction": "full_tree"
                }
             ]
          }
      }
   ]
}
```

4.2.8. GUI rules

For model forms, a set of rules can be defined that specify:

1. An initial state of the model form.
2. A change in its behavior and values on it in accordance with data and events that take place on the form.

This set of rules is defined in a GUI Rule model and it applies to a selected model's form when it is used.

GUI Rules can, for example, be used to hide or show input controls, to enter values or to enable controls in accordance with change or input on the form.

When a GUI Rule is created, the design form applies to the specified model. Field Specific rules can be specified as well as Events on fields.

If a **Field** is assigned a **Property** of `Type`, the **Value** dropdown shows possible choices. Custom values (for example `multiline`) are also allowed.

Events are associated with Actions on fields. In other words, if a certain event takes place in a field, actions can be carried out on other fields.

The actions of an event depend on conditions. More than one condition can be specified.

Note:

- In the case where a GUI Rule is applied to a Device Model, the attributes of all device versions are available for events and actions. The GUI drop-down lists for GUI Rules would list the union of device attributes. In other words, since all device versions are supported by GUI Rules, a GUI Rule can be defined to support all these device versions. If a specific form on the GUI does not display a particular field, any related GUI Rule will not be applied to the form.
- An event GUI Rule at a higher hierarchy level will precede a Field Specific GUI Rule at a lower hierarchy level, but for Dropdown Filters, we create event GUI Rules at the lower hierarchy level, thereby avoiding this precedence.

4.2.9. GUI rule for the target model of a configuration template

If a GUI Rule is created for a target model referenced in a configuration template, then the GUI Rule is applied to this target model type referenced in the Configuration Template.

For example, a GUI Rule called `ConfigurationTemplateOverride` is available for the model `device/ios/Script` (which is a target in a Configuration Template for a Script - refer to the topic on Scripts). This GUI Rule has a Field Specific rule for the field `expect_script` (originally of type String) and sets its type to `multi_line`.

GUI Rules called `ConfigurationTemplateOverride` exist for the following models:

- `device/ios/Script`
- `device/pgw/Script`

For example:

```
{
   "meta": {
       "model_type": "data/GUIRule",
       "schema_version": "0.4.9",
       "hierarchy": "sys",
       "tags": [
           "base",
           "core"
       ]
   },
   "data": {
       "field_specific": [
           {
               "field": "expect_script",
               "property": "type",
               "value": "multiline"
           },
           {
               "field": "expect_script",
               "property": "scrollbars",
               "value": "true"
           }
       ],
       "type": "device/ios/Script",
       "name": "ConfigurationTemplateOverride"
   }
}
```

4.2.10. Relations

Relations do not store data on the system. They relate groups of resource types such as device models and data models.

The purpose of a Relation is to provide a model type that can group together related models and then carry out operations on them. Model types are related by joining them similarly to a SQL “left join”. One or more fields can be specified as foreign keys.

A relation of relations is not supported.

A Relation will show all the attributes of its model types by default. Even fields specified as foreign keys will appear for each of their respective model types. Unwanted attributes are hidden using a Field Display Policy, and default values of hidden attributes can be assigned using a Configuration Template.

Operations are added to Relations. Operations with associated workflows will execute their custom workflow. *Add*, *modify* and *delete* operations without workflows will execute a dynamic workflow that simply adds, modifies, or deletes all related model instances. However, if a Relation for example contains *shared* Phones or Lines (as the Subscriber relation), then these related model instances are not deleted.

Related Topics

- Shared Lines in the Core Feature Guide.

4.2.11. Tags

During system customization and when creating dial plan models, Configuration Templates, and so on, a set of model instances can be grouped with one or more common tags. Such tagging is useful in order to find all customizations that have the same tag.

Tags can be added to an instance of a model. The tag value can be any string. (For designers: enabling the GUI Rules setting called **Show Metadata Information** for the model type will show the tags that have been added to the model instance on the model design form).

The same tag can for example be used with more than one model instance. A tag or a list of tags can be specified when a search is carried out in the user interface search box, for example `tag is feature_my_feature_name`.

See also the Search Syntax topic in the Core Feature Guide.

The same tags can for example be added to a number of model instances so that they can then all be listed by carrying out a search for this tag. The search results list is then available to carry out common tasks on the list view such as export, packaging and deletion.

4.2.12. Add and remove tags

Tags allow you to search for all model instances with a common tag so that you can perform bulk operations on model instances sharing the tag.

This procedure adds and removes tags.

Prerequisites:

- Your access profile should allow tagging on the model.
- You should have access to the model instance.

Perform the following steps:

1. Log in to the administrator interface.
2. Choose the hierarchy to which the model belongs.
3. Choose an instance of the model you wish to tag.
4. Click the **Tag** action; then:
    - To add a tag, type in a tag name.

      Important:

      – To ensure tags are searchable, use only lowercase letters in tag names.

      – Tag names should not contain spaces.

      You can add more than one tag to a model instance.

- To remove a tag:

  To remove an existing tag name; that is, <tag_name>, type the following prefix to the existing tag name: __CLEAR_TAG__<tag_name>

  To remove *all* tags, type __CLEAR_ALL_TAGS__ as the tag value.

4.2.13. Add a view

A "view" is a model type that is used to provide an input form. A view instance does not store any data entered on its form on the system. Attributes and operations (workflows or tools) on these attributes can be defined.

Users with access to this model type for customization can manage views.

1. Choose the hierarchy level for the model.
2. Open the Views list view.
3. Click **Add** on the button bar to open the Views input form.
4. Enter a Name for the Attribute.
5. Click **Add** adjacent to Attribute Properties to expand the selection.
6. Enter one or more attributes (click **Add** as required to create other instances for the same attribute) where:
   a. Name - Used as reference target.
   b. Friendly Name - Optionally an alternative field label.
   c. Description - Optionally a documentation description of the attribute that also shows as a tool tip and on the on-line help page.
   d. Data Type - Choose from the drop-down-list. The selection modifies the available attribute properties for Array and Object data types. For a String data type, a number of options are available:
      - Maximum Length.
      - Version.
      - String Format - The selected String Format acts as an input filter on the string data in the data model.
      - Is Password? - if the data type will be used to enter a password. In this case the value will be masked on forms.
      - Is Mandatory? or Is Read-Only?.

        Note: For fields that are marked read-only, these require an additional GUI Rule to provide this functionality. Since views are not designed to be updated, data in such fields would be for informational purposes.

      - Validation Regular Expression - that the string should meet. For example, "[a-z]{4}" without quotes specifies 4 characters in the range a to z in small case.
      - Default Value.

For example, the format “Date (“YYYY-MM-DD”)” requires delimited numeric text input data in the specified format and input fails if the format does not meet the format).

If the selected String Format is URI, additional controls are available to select a data-, domain- or device model (Target) attribute (Target Field) to be accessible in the created data model as a drop-down list of the Target Field attributes available in the Target model.

For data type: Anything, Boolean, Integer, Null, Number, choose if the attribute Is Mandatory? or Is Read-Only? or has a Default Value. For each required input control to be available on the form, add an **Attribute Properties** entry of the required **Data Type**.

7. Click **Add** adjacent to Choices and enter a Value and Title as required. Click **Add** to open up multiple instances if required.
8. Click **Add** adjacent to Workflows to expand the selection, and enter the required Operation and Workflow parameters. Click **Add** as required to open additional instances of the required values.
9. Click **Save** on the button bar when complete to save the created View.

4.3. Theme Customization

4.3.1. Less files and customizing themes

Overview

The preferred way to edit a theme is to edit and compile the Less files (file-based theme). The file-based theme is in the form of a Cascading Style Sheet (CSS), which is exported, edited, and re-imported to the system.

Note: Less allows you to customize a theme with minimum technical knowledge. See, for example, Twitter Bootstrap [http://getbootstrap.com/customize/>].

To find out more about Less, visit the official website [http://lesscss.org/].

This section describes the recommended practice for editing the theme *.less* files. It is however also possible to save the file, and open the CSS file in a text editor.

While all aspects of the exported CSS can be modified (for example, you can overwrite colors, sizes, fonts, and images), it is recommended that you use the default theme as a template for the basic design. This is to prevent usability and functional issues.

The CSS file is simple to export and edit. Headers in the CSS file are clearly marked, indicating the area of the Admin Portal where the design applies. The headers include these components:

- Navbar
- Shortcut menu
- Hierarchy breadcrumbs
- Quick search
- Tree menu
- Toolbar
- List

- Form

Related topics

- Introduction to Themes in the Core Feature Guide

Directory structure of theme files

Once you have downloaded and extracted a theme into a directory folder, you should see the folders and files display with the following structure:

```
.
+-- build_dependencies
+-- img
+-- skin.css
+-- skin.less
+-- variables.css
+-- variables.less
```

Important: You must maintain this directory structure.

Working with theme files

Any changes you make directly in the CSS files will have to be manually carried over after each change in the Less files.

You will need to edit the Less files and compile them to get the new CSS files.

Before you start

- Download the theme file from VOSS Automate, and extract the theme into a directory.
- You will need a Less compiler to compile the Less theme. You can find examples here:
  - Online: [http://lesscss.org/usage/index.html#online-less-compilers>]
  - On your desktop [http://lesscss.org/usage/index.html#guis-for-less]
  - IDE: [http://lesscss.org/usage/index.html#editors-and-plugins]

Image files

Custom images for Login page backgrounds and logos are added to the `img` directory.

Images can be stored with the theme or referenced with the use of relative path names. Images must be identified with the correct file path name. It is recommended that you use relative paths; that is, a path relative to the CSS file location. For example, if you created an image sub-folder called 'img', use 'img/myimage.png'. In this case, once you upload, the image is available to view at the following URL: *http://<hostname>/www/themes/mytheme/img/myimage.png*

Note: It is recommended that you do not use '/' preceding the path name.

Less files and CSS files

You can customize the following theme files:

`variables.less`	Custom variables, overriding variables set elsewhere. When you're done, this file compiles as `variables.css`.
`skin.less`	Main customization file. When you're done, this file compiles as `skin.css`.

The override hierarchy of variables and theme settings display as the first lines of `@import` instructions in the `skin.less` files, for example:

```
// Loads default variables
@import (optional) "build_dependencies/themes/minimal/variables.less";
@import (optional) "../minimal/variables.less";
// Overrides some default variables
@import "variables.less";
@import (optional) "build_dependencies/css/coreAdministratorStyles.less";
@import (optional) "../../css/coreAdministratorStyles.less";
```

Imports lower down on the list override imports higher up on the list. The `(optional)` parameter indicates the import is ignored if the file is not available.

Note: It is recommended that you do not remove any references to files in the following directory: `build_dependencies`

Customizing variables.less files

Variables are used in `skin.less` and in other `.less` files in the sub-directories of the `build_dependencies` directory.

Use `variables.less` to override any of these existing variable values. New variables can also be added if required and used in `skin.less`.

Existing variables and values are in `build_dependencies/themes/minimal/variables.less`.

Variable names have self-explanatory names and are grouped into Admin Portal categories, such as Login page, dashboard, or colors.

For example, in the default theme, show the following in `variables.less`:

```
@currentYear: ~`new Date().getFullYear()`;
@copyrightNotice: '\00A9  @{currentYear} VOSS. All Rights Reserved';

@loginLogoWidth: 200px;
@loginLogoHeight: 64px;
```

These variables are used in `skin.less`, and define:

- The footer copyright notice, for example, in `skin.css`, the `content` property:

```
body.login_page:after {
  position: fixed;
  bottom: 0;
  content: '\00A9  2017 VOSS. All Rights Reserved';
  left: 0;
  color: #eeca;
  z-index: 1;
  background-repeat: no-repeat;
  background-position: left;
}
```

- The dashboard logo image size HTML attributes:

```
<img style="user-select: none;"
 src="http://localhost/www/themes/default/img/login_logo.png"
 width="200"
 height="64">
```

Customizing skin.less files

To customize `skin.less` files, override entries in the imported `.less` files, found in the sub-directories of `build_dependencies`.

In the compiled `.css` file, override theme changes will then follow the original CSS entries.

It is recommended that you enable browser developer tools so that it's possible to identify matching HTML properties of portions of the Admin Portal, as well as the associated Admin Portal styling in the `skin.css` file.

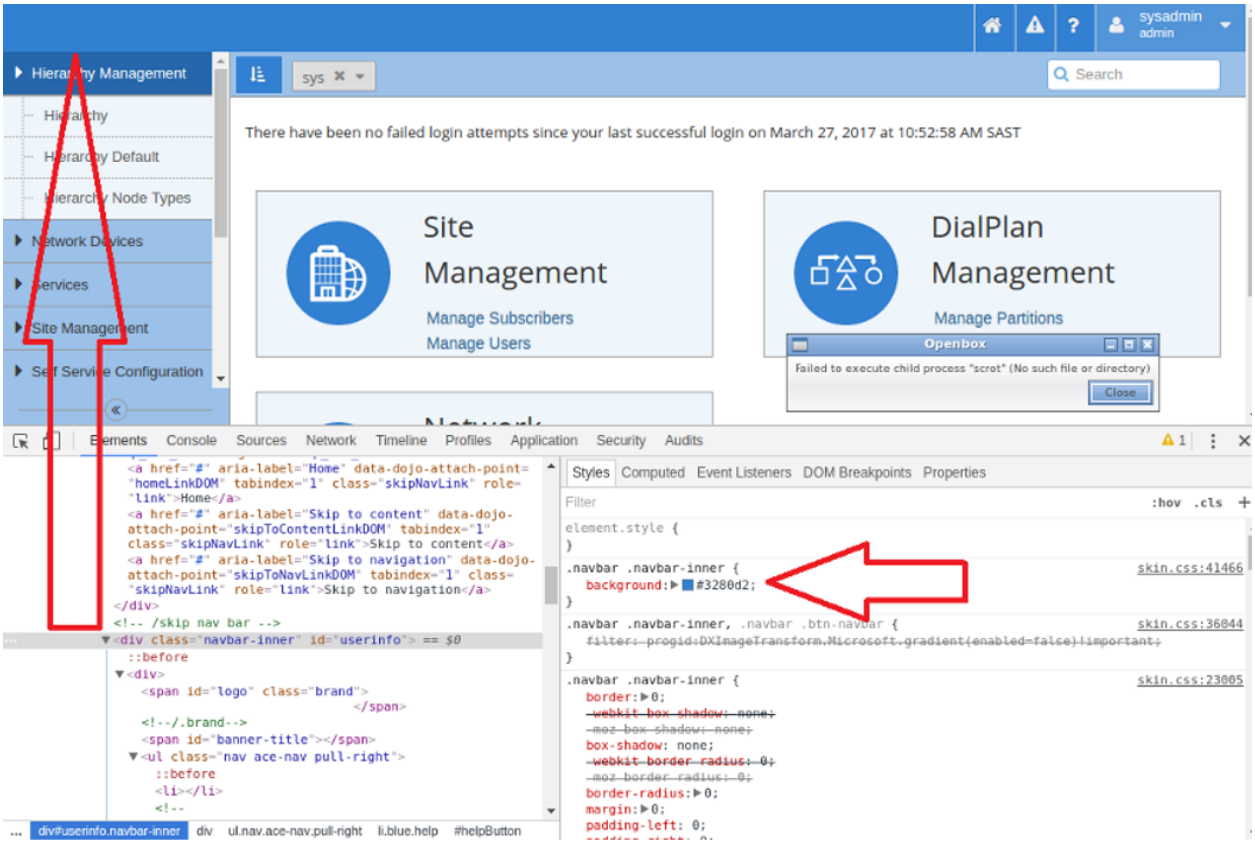
Example

Consider an entry in the default theme `skin.less` under the heading `Navbar`. The Navbar theme settings apply to the navigation headers of the application: the top and side menu bars. For more details on the navbar component, see: Navbar.

```
.navbar .navbar-inner {
  background: @mainBackgroundColor;
}
```

In this case, the background color of the horizontal navbar is modified. The value of `@mainBackgroundColor` is defined earlier, as `#3280d2`.

Use the browser developer tools to inspect how this override applies to the theme:



In the compiled `skin.css` file, the update in the snippet from the `.less` file above will follow the main definition:

```
.navbar .navbar-inner {
  background: #3280d2;
}
```

The main definition shows all the values that apply to `navbar-inner`. The value of `background-color` is overridden.

```
.navbar-inner {
    min-height: 40px;
    padding-right: 20px;
    padding-left: 20px;
    background-color: #fafafa;
    background-image: -moz-linear-gradient(top,#fff,#f2f2f2);
    background-image: -webkit-gradient(linear,0 0,0 100%,
      from(#fff),to(#f2f2f2));
    background-image: -webkit-linear-gradient(top,#fff,#f2f2f2);
    background-image: -o-linear-gradient(top,#fff,#f2f2f2);
    background-image: linear-gradient(to bottom,#fff,#f2f2f2);
```

(continues on next page)

(continued from previous page)

```
    background-repeat: repeat-x;
    border: 1px solid #d4d4d4;
    -webkit-border-radius: 4px;
    -moz-border-radius: 4px;
    border-radius: 4px;
    filter: progid:DXImageTransform.Microsoft.gradient(
      startColorstr='#ffffffff',
      endColorstr='#fff2f2f2',GradientType=0);
    *zoom: 1;
    -webkit-box-shadow: 0 1px 4px rgba(0,0,0,0.065);
    -moz-box-shadow: 0 1px 4px rgba(0,0,0,0.065);
    box-shadow: 0 1px 4px rgba(0,0,0,0.065)
}
```

Similarly, other values in the CSS file for `.navbar-inner` can be overridden.

Example

If the alert notification pop-up style (for example with timeout notifications) should also be customized to match the custom style, you can add the following style (with suitable style colors):

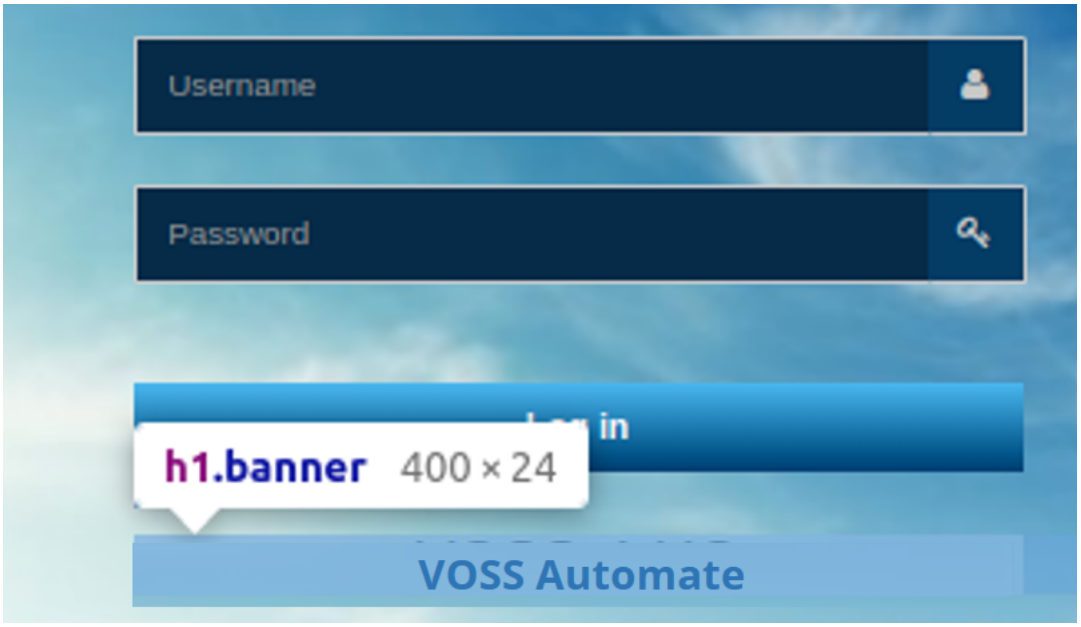
```
.alert-info {
  color: #31708f;
  background-color: #d9edf7;
  border-color: #bce8f1;
}
```

4.3.2. Theme banner customization

For Less files in general and how to update, compile and upload your theme, see *Less files and customizing themes*

To customize the look and feel of the banner text managed from the theme, inspect the styles in `skin.css` that apply to the theme Site Title and Banner Text. Both are marked up in HTML as `<h1>`:

- Site Title: `.login_page h1`
- Banner Text: `h1.banner`



If a style is defined for the Site Title and there is no style override for the Banner Text, then the Site Title style applies to the Banner Text. More generally, styles for `h1` apply.

For example, unless the `h1.banner` customization includes values for `font-size` and `color`, the values of the variables in `variables.less`: `@loginBannerFontSize` and `@loginBannerFontColor` will apply to the banner text.

In `skin.less`, Banner Text style customization can for example include the following attributes,

```
h1.banner {
    color:
    font-family:
    font-size:
    font-weight:
    line-height:
    margin:
    margin-left: [use negative value to show text outside login form box] - see below]
    text-align:
    text-shadow:
    width:
}
```

For `margin-left`, if you wish to center the banner, the negative margin and total width will have to be calculated. The `margin-left` value should be the negative of:

`(banner width - login form width)/2`

For example, if the username/password fields are `400px` wide (determine this with the browser developer tools), and the banner width is set to `800px`, then the `margin-left` of the banner should be:

`(800px-400px)/2 = -200px`

Therefore, the `skin.less` file will contain:

```
h1.banner{
  width: 800px;
```

(continues on next page)

(continued from previous page)

```
  margin-left: -200px;
 ...
}
```

4.3.3. Privacy policy menu items

VOSS Automate supports management of privacy policy notices on the user interface. This allows compliance with General Data Protection Regulation (GDPR) requirements.

By default, high level system administrators above the Provider level hierarchy can manage privacy policy references that are available on user menus. These administrators can provide the required access to the `data/PrivacyPolicy` data model and add menus to lower level administrators if required.

Privacy policy references can be set up for each hierarchy. If one is not added to a specific hierarchy, the one at the next higher hierarchy applies.

When a privacy policy applies to a user hierarchy:

- On the Admin Portal, a privacy policy menu item is added to the bottom of the user's menu. The title of the menu item is the name of the created policy.
- On the Self-service GUI (if available), a side button bar menu item is added. The title of the menu item is **Privacy Policy**.

When selecting the menu item, the link URL of the policy opens on a new browser tab.

Note:

- For the Admin Portal, the Privacy Policy menu item is not visible from a menu layout and cannot be managed from **Menu layouts**.
- Login page privacy policy links are managed from **Themes**.

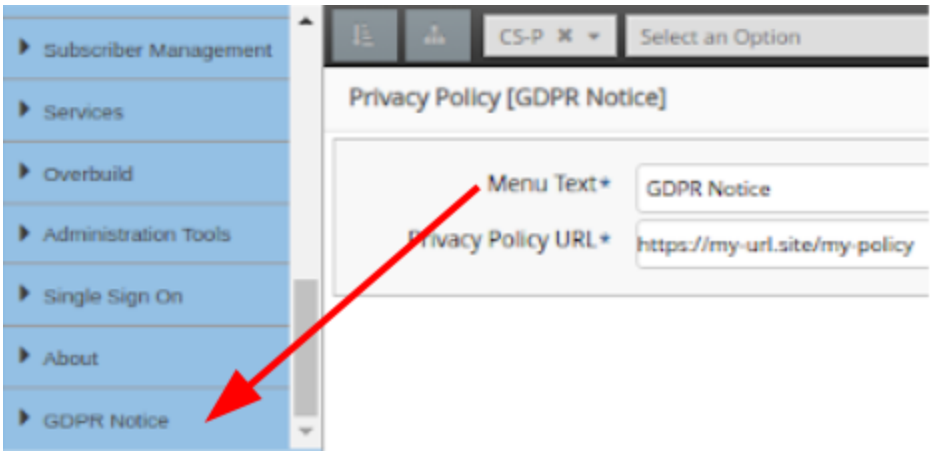
4.3.4. Manage privacy policy menu items

Tip: *Use the Action search to navigate Automate*

---

1. Log in as an administrator with the required privacy policy management permissions and menu access.
2. Go to the page, for example by default, **Privacy Policy Configuration**. The list view shows privacy policy names and links at various hierarchies in the system. Privacy policies can then be added, modified and deleted.
3. To add a privacy policy, navigate to the hierarchy at which the privacy policy should be added and click **Add**.
4. Add a Name, Privacy Policy URL and click **Save**. Note that this name becomes the menu item name.

On the Admin Portal, a privacy policy menu item is added to the bottom of the user's menu - for users at the specified hierarchy or lower and without a privacy policy on their own hierarchy. On the Self-service GUI, a side button bar menu item is added.



4.3.5. Manage themes

Tip: *Use the Action search to navigate Automate*

Overview

You can use the **Themes** page in the Admin Portal to create or modify a theme.

You can select the following tabs on the **Themes** page in the Admin Portal:

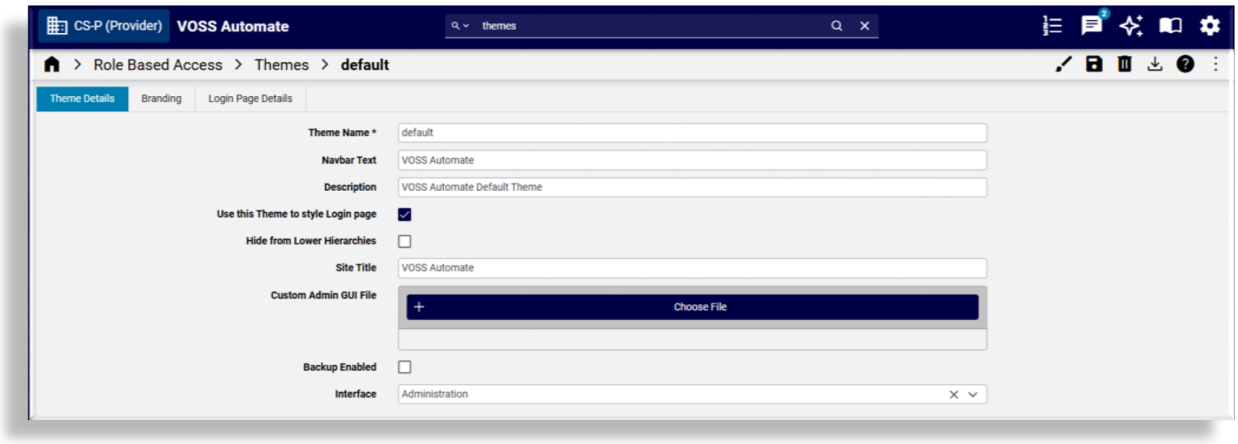
- Theme Details
- Branding
- Login Page Details

Theme settings

Theme details tab

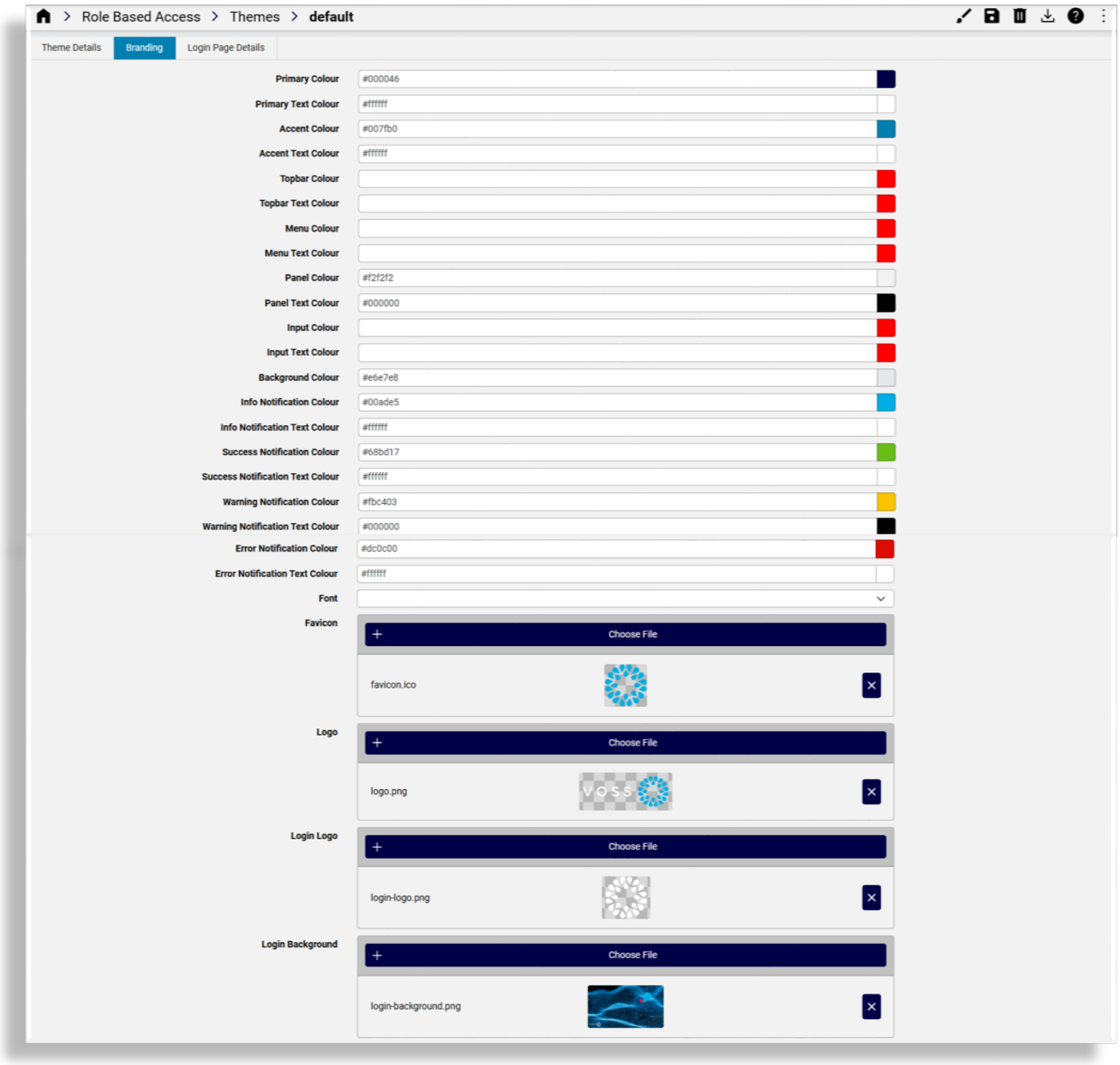
On the **Theme Details** tab of the **Themes** page you specify theme details:

- Theme Name: Mandatory. The name of the theme. Valid characters are in the range of `a-zA-Z0-9_`.
- Navbar Text: Text to display on the navigation bar, adjacent to the organization level.
- Description
- Whether to use the theme to style the Login page.
- Whether to hide the theme from lower hierarchies.
- Site Title: The title of the site, displays on the browser tab
- Custom Admin GUI File: Upload a custom theme file, if applicable
- Whether to enable a backup of the theme (images and other custom elements)
- Interface: The interface where the theme is applied (Administration or Self-service)



Branding tab

On the **Branding** tab you can customize a theme for your organization. You can change colors via the color picker or by typing in the color hex value. When no colors are chosen in this tab, the defaults apply.



Fonts

The **Font** dropdown allows for a selected font to override the global default **Roboto** font. The selected font is then applied to all text, including the login screen.

Uploading images

When uploading images for the theme:

- Note file size and *width x height* pixel dimension size restrictions. A system message displays if the image is too large.
- Only PNG files are supported for the Logo image. Other images can be PNG or JPEG.
- For image filenames, you can use the following characters and character types:

  ```
  ALPHA / DIGIT / "-" / "." / "_" / "~"/ "#"
  ```

Image details

- **Favicon**: The favicon for the site. Shown on the tab and when the site is bookmarked.
  - Type: PNG image or with `.ico` extension
  - Maximum dimensions: 256x256 pixels
- **Logo**: This image is used for the logo in the top left of the menu bar.
  - Type: PNG image with a transparent background
  - Maximum file size: 0.5MB
  - Maximum dimensions: 600 pixels in width and 192 pixels in height
- **Login Logo**: This image is used for the logo on the login page
  - Type: PNG image with a transparent background
  - Maximum file size: 0.5MB
  - Maximum dimensions: 600 pixels in width and 192 pixels in height
- **Login Background**: This image is used for the login screen background
  - Type: PNG or JPEG image
  - Maximum file size: 5MB
  - Maximum dimensions are 1920 pixels in width and 1080 pixels in height
- **Menu Background**: This image is used for the side menu background
  - Type: PNG or JPEG image.
  - Maximum file size: 2MB
  - Maximum dimensions: 240 pixels in width and 1040 pixels in height.

Login page details tab

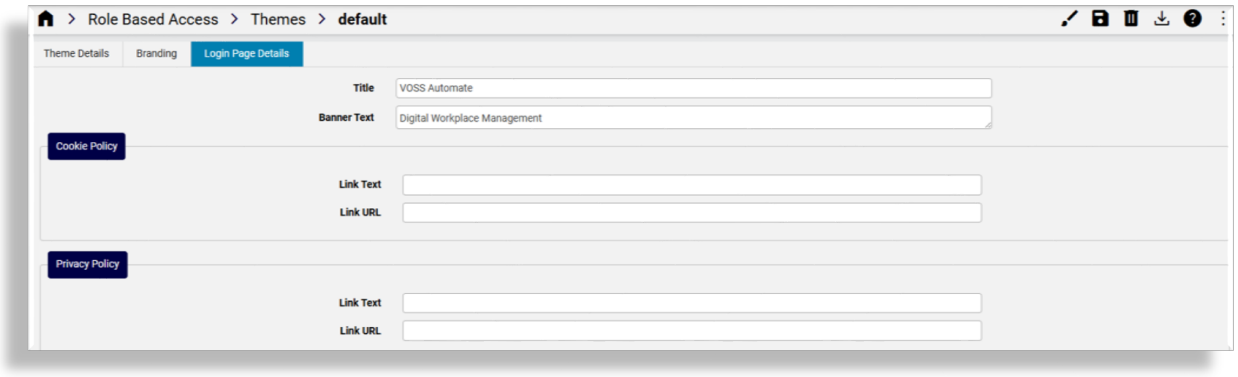
The **Login Page Details** tab defines the theme for the Login page, including the title and banner text, cookie policy and privacy policy details.

If you add banner text (limited to 2048 characters), this is used at the bottom of the Login page.

References to the cookie policy and privacy policy in the Banner Text field should be added as placeholders, which will then resolve to the **Cookie Policy** and **Privacy Policy** data entered. The placeholders are:

- `{{cookie_policy}}`
- `{{privacy_policy}}`

Note: You can add multiple lines for the banner text, including paragraphs. Banner text displays exactly as you add it to this field. Cookie and security references show as links that open in a new browser tab.



Custom themes

You can create a custom theme to change the following properties of the Admin Portal:

- Primary and accent colors
- Logo image
- Login screen background image

  If the background image also contains logos, it is recommended that these be placed on the bottom of the image.
- Background image for menu
- Browser tab title

Themes created in the Admin Portal can't be exported in full.

Preview a theme

When creating a theme, you can use the toolbar **Preview** icon (Paintbrush) to preview your theme look and feel (colors and images) before assigning it to a role and having it apply to the GUI.

4.3.6. Theme element color references for the Admin portal

Tip: *Use the Action search to navigate Automate*

Note:

- Color selection on the **Branding** tab of a theme *always* affects the Admin Portal.

  To edit and manage theme files:

  Refer to Less files and Theme Customization in the Advanced Configuration Guide.
- Color selection is optional. Where no colors are selected, defaults apply.
- If manual input of color Hex values is required, ensure the value is prefixed with #.

Admin portal default colors

Default color reference table:

Title	Field Name	Default Value (Hex)	Notes
Primary Color	primary_colour	#000046	This is the background color for most menus and headers, as well as the text color for links and buttons.
Primary Text Color	primary_text_colour	#ffffff	This is the text color for anything with the primary color background.
Accent Color	accent_colour	#007fb0	This color is used when attention needs to be drawn for important notifications or active buttons and text.
Accent Text Color	accent_text_colour	#ffffff	This is the text color for anything with the accent color background.
Topbar Color	topbar_colour	#000046	The color used for the top bar of the site. Will use the primary color if no value is given.
Topbar Text Color	topbar_text_colour	#ffffff	This is the text color for the top bar. Will use the primary text color if no value is given.
Menu Color	menu_colour	#000046	The color used for the menu on the left. Will use the primary color if no value is given.
Menu Text Color	menu_text_colour	#ffffff	This is the text color for the menu. Will use the primary text color if no value is given.
Panel Color	panel_colour	#f2f2f2	The color used for all the panels in the app.

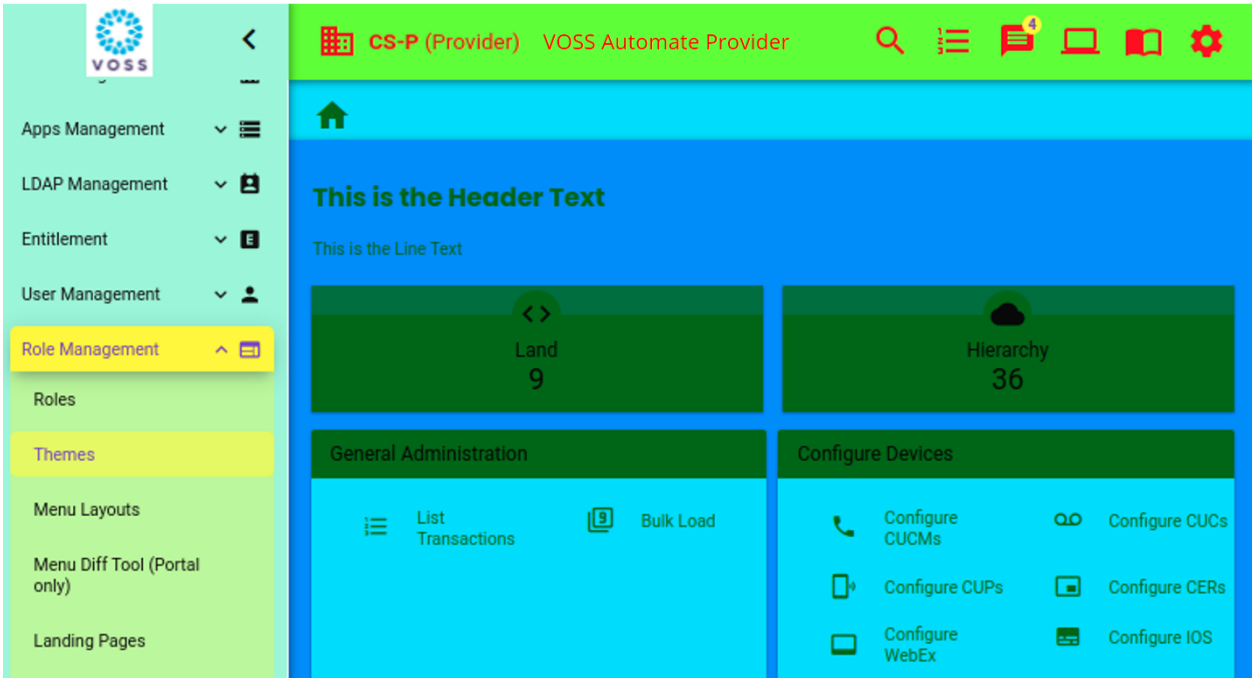
Title	Field Name	Default Value (Hex)	Notes
Panel Text Color	panel_text_colour	#000000	This is the text color for normal text in the app.
Input Color	input_colour	#ffffff	The background color for input fields. Will use the panel color if no value is given.
Input Text Color	input_text_colour	#414042	The text color for input fields. Will use the panel text color if no value is given.
Background Color	background_color	#e6e7e8	The color of the background behind panels.
Info Notification Color	info_notification_colour	#00ade5	The color used for info notifications.
Info Notification Text Color	info_notification_text_colour	#ffffff	This is the text color for info notifications.
Success Notification Color	success_notification_colour	#68bd17	The color used for success notifications.
Success Notification Text Color	success_notification_text_colou	#ffffff	This is the text color for success notifications.
Warning Notification Color	warn_notification_colour	#fbc403	The color used for warning notifications.
Warning Notification Text Color	warn_notification_text_colou	#000000	This is the text color for warning notifications.
Error Notification Color	error_notification_colour	#dc0c00	The color used for error notifications.
Error Notification Text Color	error_notification_text_colour	#ffffff	This is the text color for error notifications.

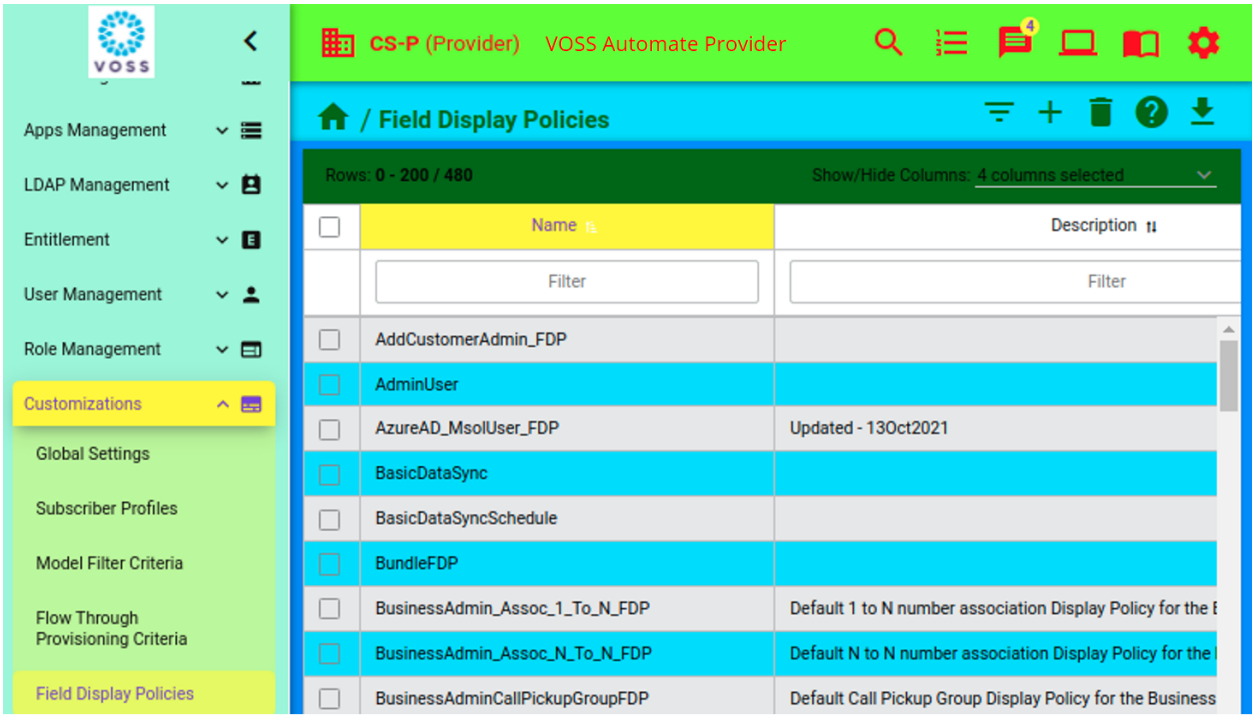
On the Admin portal, consider the color selection on the **Branding** tab:

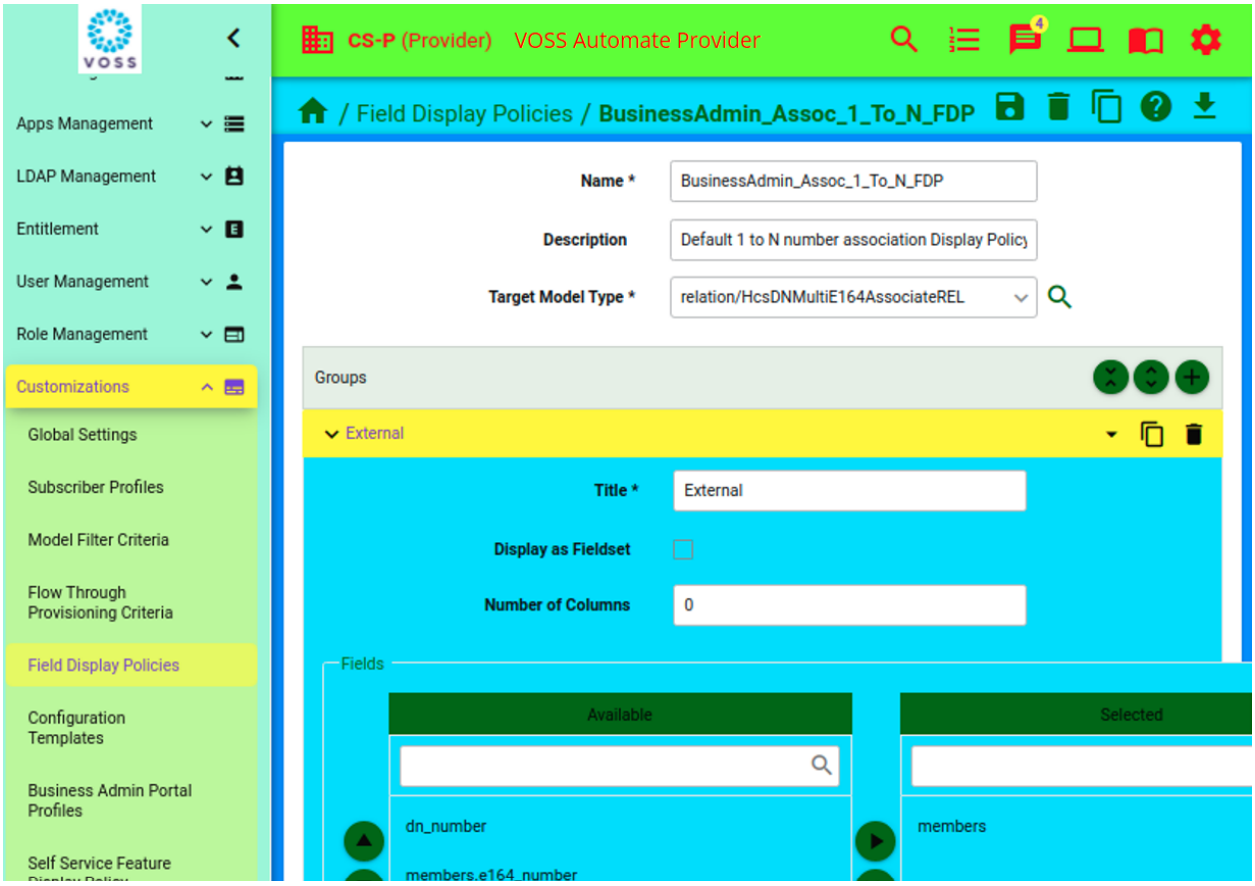
Note:

- For details on images and logos, see: *Manage themes*.
- If a color value appears blank, default values apply.
- Sub-menu and sub-sub-menu backgrounds are rendered as percentages of the Menu Color.

Admin portal







4.3.7. Automate cookie policy

When formulating a cookie policy, customers should include details on the use of cookies by Automate. The following text provides details on the use of cookies in Automate that can be included in the policy:

```
VOSS Automate uses cookies for the following purposes:

Personalisation - we use cookies to store information about your most recent settings,
preferences and to personalize our website for you.

The cookies used for this purpose are:

hierarchyTreeSaveStateCookie
resourceTreeSaveSelectedCookie
resourceTreeSaveStateCookie
ace.settings
sso_login_url

Security - we use cookies as an element of the security measures used to protect
user accounts, including preventing fraudulent use of login credentials, and to
protect our website and services generally.

The cookies used for this purpose are:
```

(continues on next page)

(continued from previous page)

```
* Administrator login:

  csrftoken
  sessionid

* Self-service login:

  csrftoken
  sessionid
  session
  rbacInfo
```

4.4. Menu Customization

4.4.1. Menu diff tool

System level (hcsadmin, entadmin) and provider administrators have access to the Menu Diff tool on the menu to allow for a side-by-side comparison and management of two selected menu layouts:

- **Source Menu**: a drop-down list of menu layouts from hierarchies at the administrator's hierarchy *and higher*.
- **Target Menu**: a drop-down list of menu layouts from hierarchies at the administrator's hierarchy *and lower*.

The side-by-side forms with the two selected menus can be expanded using the **Expand**/**Collapse** button.

Menu differences are highlighted as follows:

- The same menu item in both menus but on a different menu path is highlighted.
- A menu item in one menu but not in the other menu is highlighted.

To update the target menu, drag and drop menu items from the source menu to a position in the target menu.

- Where a menu item is now in both menus, it is not highlighted anymore.

Note:

- All menu properties (e.g. Field Display Policy, model reference, and so on) are copied.
- If the menu item than is copied contains sub-menus, these are included.

Click **Save** to save changes in the target menu.

Related topics

- Menu Merge Tool in the Core Feature Guide

4.5. On-line Help

4.5.1. Online help customization

Note: The customization to on-line help discussed here refers to the data in the `data/GeneralHelp` model.

The online help information that is available can include general and field-level information. This information can be added in the following ways:

- On the design input form of the model. In this way the creator of a model can include its on-line help information. Field-level help is also shown as a tool-tip pop-up in the instance input form of the model.
- In a Field Display Policy that applies to the model. In this way the default information on the model can be overridden if the Field Display Policy is applied to the model. A Field Display Policy can be associated with a model from its access in the Menu Layout.
- If write access to the `data/GeneralHelp` model is available, instances of this model can be marked as information to display on the on-line help for a model. This is done by adding Model Overwrite Details array instances to a `data/GeneralHelp` model instance.

Role-based access on models and model fields, as well as Field Display Policies for models can be used to customize a user's view of the available on-line help.

Instances of the `data/GeneralHelp` model created at a selected hierarchy level provide hierarchy-specific content. If the name of the instance of `data/GeneralHelp` at a hierarchy level is the *same* as that of an existing default instance, the default instance is replaced at the relevant hierarchy level.

Instances of the `data/GeneralHelp` model also contain a language code. If the user's language corresponds with the language code, then this instance is displayed. In the case where an instance is to be translated, a new instance can be created where:

- The instance `language_code` is updated to the new language, for example `fr-fr`.
- Instance names should be unique.

If write access to the `data/GeneralHelp` model is available (contact VOSS), new help instances can be added and existing help instances can be modified. The process is summarized in the list below:

- The menu position of the instance (the value `-1` will hide an instance). The list of instances can be sorted on the menu position column to show the order in which they will display on the General Help list on the Help browser tab.
- The display of an instance as a part of the model detail help of one or more selected models by adding **Model Overwrite Details** array instances to a `data/GeneralHelp` model instance. Options are available to:
  - Modify the title of the model help.
  - Modify the title of the General Help instance.
  - Show or hide the model help information (as obtained from the model or Field Display Policy).
  - Specify the position of the General Help instance relative to the model help information.

If more than one General Help instance is set to override the *same* selected model, *all* these overrides are applied where the positioning does not conflict. If positioning conflicts, the positioning of the *first* of the General Help instances (sorted alphabetically by instance name) is used.

If formatting is required in a new General Help model instance, the content can be written as ReStructured Text (http://en.wikipedia.org/wiki/ReStructuredText).

Note that:

- The Restructured Text is added as a *single string* in the JSON format file (line breaks in the text are to be replaced by the JSON escaped equivalent). This will be seen in the file when the `data/GeneralHelp` instance is exported.
- Model descriptions are also pop-up tooltips, so Restructured Text is not supported in attribute descriptions - either in the model or Field Display Policy.
- There is no support for local image references, but remote references to images can be added in the Restructured Text by adding the URL to it.

4.5.2. Online help model override options

When adding or modifying an instance of the online help, the Model Override Options tab in the Admin Portal on the instance view provides settings to customize the context-sensitive help display.

Setting	Description
Hide Model Help Title	Deprecated
New Model Help Title	Deprecated
Insert General Help Text Into Model Content	If Manipulate Model Dynamic Generated Help is enabled, this option also toggles the insert of the data/GeneralHelp instance text content.
Hide General Help Section Title	If enabled, the data/GeneralHelp instance Menu Title or else Name is displayed.
New General Help Section Title	By default, the RST title of the data/GeneralHelp instance is shown. This title can be replaced by entering text here.
New Model General Help Paragraph Position	The position of the current data/GeneralHelp instance relative to the dynamically generated content is specified by an integer. If more than one instance of data/GeneralHelp will be applied to the same model type, the ordering of instances is set here.
Manipulate Model Dynamic Generated Help	This setting is enabled to override the dynamically generated online help page with all the content and settings here.
Model Dynamically Generated Help Paragraph Position	The position of the dynamically generated content relative current data/GeneralHelp instance to the is specified by an integer. If more than one instance of data/GeneralHelp will be applied to the same model type, the position of the dynamically generated instance is determined here.
Hide API Help Link	The display of the default link to the API reference on the dynamically generated help page can be toggled.
Hide Visualize Button	Deprecated
Affected Model Type	Select the model to which the override applies. More than one override instance can be added so that the content of the data/GeneralHelp instance can override more than one model.

4.6. Scripts

4.6.1. Scripts

It is possible to create and execute SSH scripts on devices from within the system. Scripts can be executed as part of a provisioning workflow.

Script files syntax is the Expect script syntax with the ability to add one of the following per line of the script:

- send data to a device and evaluate and substitute macros visible in the context hierarchy
- define a regular expression for the response expected from the device
- branch to a next line based on a match to a response from a device
- comments can be added to the script

Scripting has been tested with Cisco IOS devices.

Please note:

- Branching is only for one line in the script file and does not yet support blocks of commands.
- Security: There is no control of what can be done or not done on the device.

In order to create and run a script, a supporting data model and IOS script device model Configuration Template can be used as a part of a Provisioning Workflow that is executed.

A GUI Rule called `ConfigurationTemplateOverride` for `device/ios/Script` is applied by default to the Configuration Template, which adds a `multiline` property to the `expect_script` field of the device model. This enables multiline input of the script.

The supporting model for the IOS device is `data/Ios`, to which an instance is added containing the host, port and authentication details of the device. This instance can then be selected as the Network Device Filter in the Provisioning Workflow if required. Alternatively, the Provisioning Workflow can define a context variable, say `ios_details`, that resolves to a particular `data\Ios` instance, for example `{{data.Ios.* || direction:local}}`

In the IOS device model Configuration Template, the script itself is entered and can consist of a combination of Expect script syntax and system macros that are for example used for standard script blocks or variables for the selected `data/IOS` instance values.

When the Provisioning Workflow is executed, the transaction log can be inspected to examine the entire script with macros and variables referenced, as well as the device response.

4.6.2. Add and run a script

1. Add an instance to `data/IOS`. This instance contains the host, port and authentication details of the device.
2. Create a Configuration Template for the Target Model Type selected as `device/ios/Script`. A GUI Rule called `ConfigurationTemplateOverride` is applied by default to it to enable multiline input.

   Macros in a script should however be written on one line.

   In the device/ios/Script group on the Configuration Template, the following values can be entered:

   - The Description value can be entered.
   - The Expect Script value is the entered. The script can be entered over multiple lines. Variables can be used in the entered script - sourced from either context variables or macros. Refer to the Expect Script Examples topic.
3. Add a provisioning Workflow.
   - Select the Workflow Operation as run.
   - Select the Step Type as model.
   - Select the Entity or Workflow as `device/ios/Script`
   - The Network Device Filter group's Device Instance can be a selected `data\Ios` instance, or if not, the device can be the value of a Provisioning Workflow Context variable that resolves to a particular `data\Ios` instance, for example `{{data.Ios.* || direction:local}}`.
   - The Configuration Template is the created template that contains the details of the script. If a Device Instance was added to the Provisioning Workflow, the script can reference it with the variable `device_details`, for example the `data/Ios` instance as `{{device_details.host}}` and `{{device_details.host}}`.

- To run the script, execute the workflow.

4. Inspect the transaction. The details of the transaction show the script with all macros and variables resolved, as well as the response from the device.

4.6.3. Expect script examples

This section shows some examples of scripts entered into a Configuration Template for `device/ios/Script` that can be used in a Provisioning Workflow to run the script. The examples show the usage of macros and variables.

Consider the following script snippet:

```
spawn telnet {{ pwf.ios_details.host }}  23\n
```

If the Provisioning Workflow from which the script is executed contains a context variable called `ios_details` with a value called `{{data.Ios.* || direction:local}}`, then the device host will be the value of the `data.Ios` instance in the current hierarchy.

Consider the following script snippet:

```
send "ping {{data.Countries.ios-country_code | country_name: 'South Africa'}}\r"\n
```

The script will be:

```
send "ping ZA\r"\n
```

Consider the following multi-line macro called `IOS_enable`:

```
expect "HQ>\r"
send "enable\r"
expect "Password:\r"
send  "{{device_details.enable_password}}\r"
```

The macro in a multiline script needs to be entered on a single line - without line breaks.

If we had macros for other snippets called `IOS_login` and `IOS_show_clock`, then the Configuration Template Expect Script value can be entered as these three macros:

```
{{IOS_login}}{{IOS_enable}}{{IOS_show_clock}}
```

4.6.4. Expect script import format

This section shows examples of import files containing multiline scripts. The purpose of these examples are to show the format of the scripts in such files.

The JSON file format shows the line break characters (`\n`) in the multiline script. Note that the example here shows `expect_script` with three line breaks for display purposes.

```
{
 "meta": {},
 "resources": [
   {
     "meta": {
       "model_type": "data/ConfigurationTemplate",
       "pkid": "[pkid]",
       "schema_version": "0.1.8",
       "hierarchy": "sys",
       "tags": []
     },
     "data": {
       "target_model_type": "device/ios/Script",
       "name": "Multiline Test",
       "merge_strategy": "additive",
       "template": {
         "description": "Multiline Test",
         "expect_script": "ssh\nhost\npassword\n{{data.Countries.iso_country_code |
           country_name:'South Africa'}}\nping\nping {{data.Countries.iso_country_code |
           country_name:'South Africa'}}"
       }
     }
   }
 ]
}
```

A MS Excel sheet cell that contains a multiline script would display in the spreadsheet editor across multiple lines (Using <Alt>-<Enter> for line breaks in MS Excel). The content of the cell would then display as:

```
ssh
host
password
{{data.Countries.iso_country_code | country_name:'South Africa'}}
ping
ping {{data.Countries.iso_country_code | country_name:'South Africa'}}
```

These formats can also be obtained by exporting an existing expect script in either JSON or Excel.

5. Move Customizations (Provider)

5.1. Network device list selection rules advanced configuration

NDL popups are controlled by GUI Rules at hierarchy levels for model types. The device selection given GUI Rules, NDLs, NDLRs and Device Selection Rules are shown in this table.

GUI Rule	NDL(s)	NDLR	Use Popup	Use NDLR	Expected Result
N	N	N	•	•	Normal Device selection
N	Y	N	•	•	Normal Device selection
N	Y	Y	•	•	NDLR is used as target device
Y	Y	N	N	N	Normal Device selection and override with NDF in workflows
Y	Y	Y	Y	N	Pop up list of NDLs
Y	Y	Y	Y	Y	Pop up list with NDLR as only option
Y	Y	N	Y	Y	Pop up an empty list with NDLR missing message
Y	N	N	N	N	Normal Device selection and override with NDF in workflows
Y	Y	Y	N	N	NDLR is used as target device
Y	Y	Y	N	Y	NDLR is used as target device
Y	Y	N	Y	N	Pop up list of NDLs (Most popular option)

The Rule Model Device Selection Type model also provides this functionality:

- The NDL device meta is available to the context in Provisioning Workflows. For example:

```
"device_meta": {
   "ndl": {
      "name": "NDL1",
      "pkid": "54dc76c82afa4327de0d218e",
      "data/CallManager": {
```

(continues on next page)

(continued from previous page)

```
         "pkid": "54dc76c72afa4327de0d217f",
         "bkey": "[\"10.120.2.175\", \"8443\", \"P.C\"]"
      },
      "bkey": "[\"NDL1\", \"P.C\"]",
      "data/UnityConnection": {
         "pkid": "54dc76be2afa4327de0d210b",
         "bkey": "[\"172.29.41.72\", \"443\", \"P.C\"]"
      }
   }
}
```

- NDL device meta namespace device_meta is available in macros as: {{ device_meta.???}}, for example:

```
device_meta.ndl.name
device_meta.ndl.data/CallManager.pkid
```

- The [ndl] macro is available for use in GUI Rules - similar to [hierarchy].
- An API parameter is available for the selected NDL when a GET request is sent for the Add form of a Relation. The value of [ndl] in the example below is a valid PKID for the NDL. For example:

```
GET /api/v0/relation/UswerCucmCucRel/add/?
    hierarchy=[hierarchy]&
    ndl=[ndl]&
    schema=true&
    schema_rules=true
```

This parameter is transformed in the subsequent Add calls to devices to a device parameter.

6. LDAP Sync

6.1. Change LDAP user sync

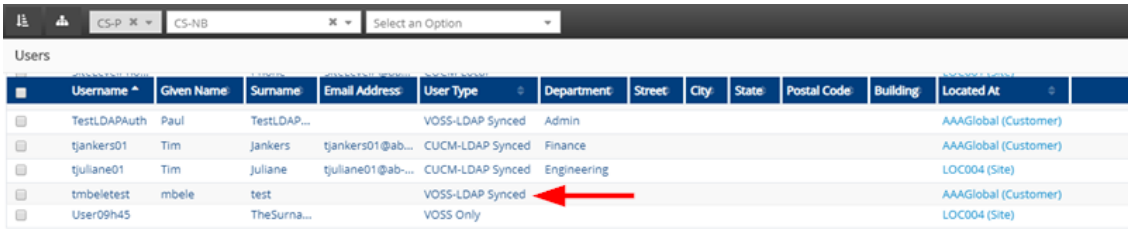
6.1.1. Overview

In top-down LDAP user management, LDAP users are first added to VOSS Automate and then synced to Cisco Unified Communications Manager (CUCM).

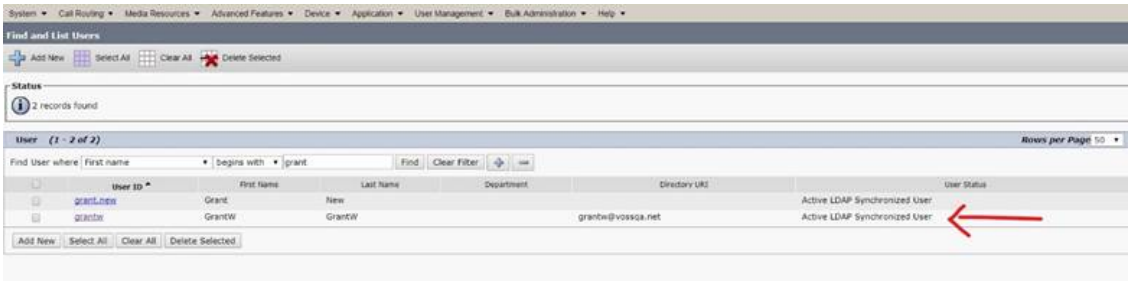
This procedure describes how to change LDAP user sync from top-down to bottom-up, that is, LDAP users on CUCM are synced to VOSS Automate.

6.1.2. Before you start

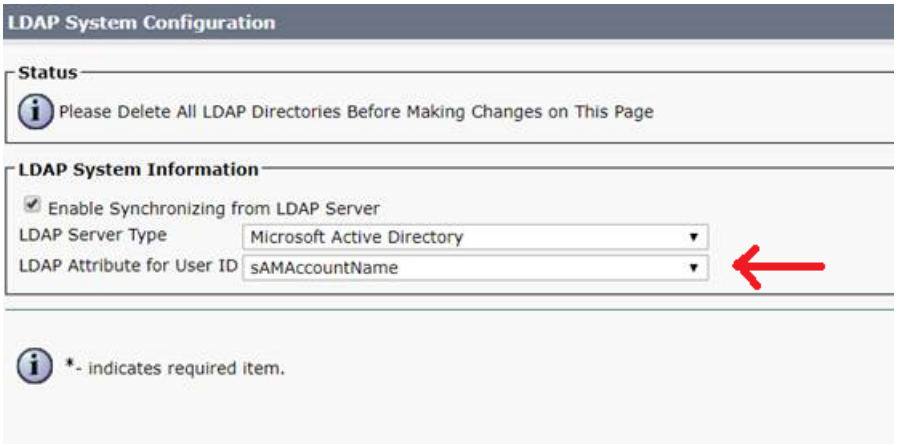
1. Prepare for the change:
   - Take a VM snapshot.
   - Ensure that the LDAP server is in sync with Automate and that Automate is in sync with CUCM.
   - Ensure that you have the correct LDAP server information.
   - Ensure that Cisco and VOSS (L3 support) are notified of this change before commencing.
   - Always first test the procedure for one user only, using a model instance filter. Contact VOSS support in the following cases:
     – If the model instance filter is to apply to the top-down LDAP to VOSS Automate synced user, it should be on the `device/ldap/user` and the attribute `cn`. You can obtain the `cn` from the LDAP synced users list.
     – If the model instance filter is to apply to the bottom-up, CUCM to VOSS Automate synced user, it should be on the `device/cucm/user` and the attribute `userid`.
2. Check the following:
   - The **Users** list in Automate shows the user is “VOSS-LDAP Synced”, and on the **Provisioning Status** tab for the user, the user is synced with both LDAP and CUCM.

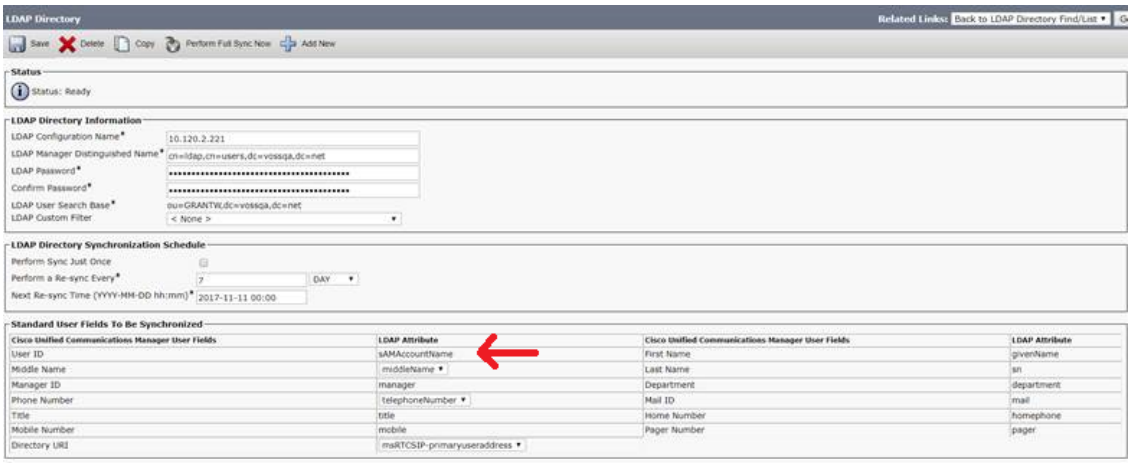


- The **User Status** column for the user in CUCM displays “Active LDAP Synchronized User”.



- The LDAP server is configured on CUCM and the **LDAP Attribute for User ID** is the same as the **Login Attribute Name** on VOSS Automate. (On CUCM: **System > LDAP > Server** and **System > LDAP > LDAP Directory** and search to find it or add it.)





- In the Automate schedules and transactions, confirm that recent LDAP - Automate syncs have occurred, and that CUCM has the same user count as Automate.

- In Automate, ensure that on **LDAP Management > LDAP User Sync** the user modes for Move, Delete, and Purge are set to “Manual”. Saving this configuration triggers a full LDAP sync.

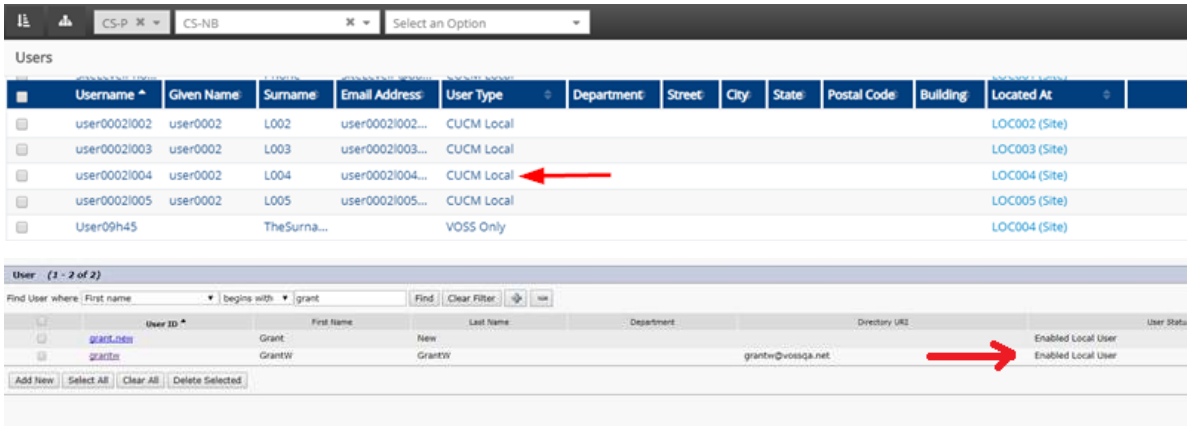
3. Make backups of LDAP server and configurations in Automate. The recommendation is to export to JSON data from the following menu paths:
   - **LDAP Management > LDAP Sever**
   - **LDAP Management > LDAP User Sync**
   - **Administration Tools > Scheduling**, LDAP Sync schedule
   - **LDAP Management > LDAP Authentication Users**

Note: Exporting to JSON data is done for troubleshooting in case of errors. However, export is limited to 200 at a time, so for a customer with for example a 5000 user count, this is impractical. In that case a VM snapshot is recommended.

6.1.3. Change the user sync from top-down to bottom-up

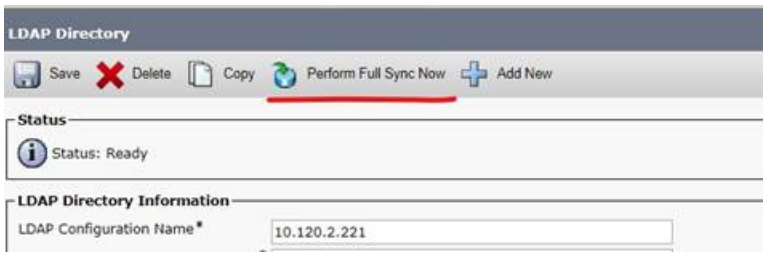
Ensure you have performed the tasks to prepare for this change, then, to change the LDAP user sync from top-down to bottom-up:

1. In Automate, remove the instance under **LDAP Management > LDAP User Sync** for this customer.
2. Verify that the relevant users display as local users on both VOSS Automate (“CUCM Local”) and CUCM (“Enabled Local User”).

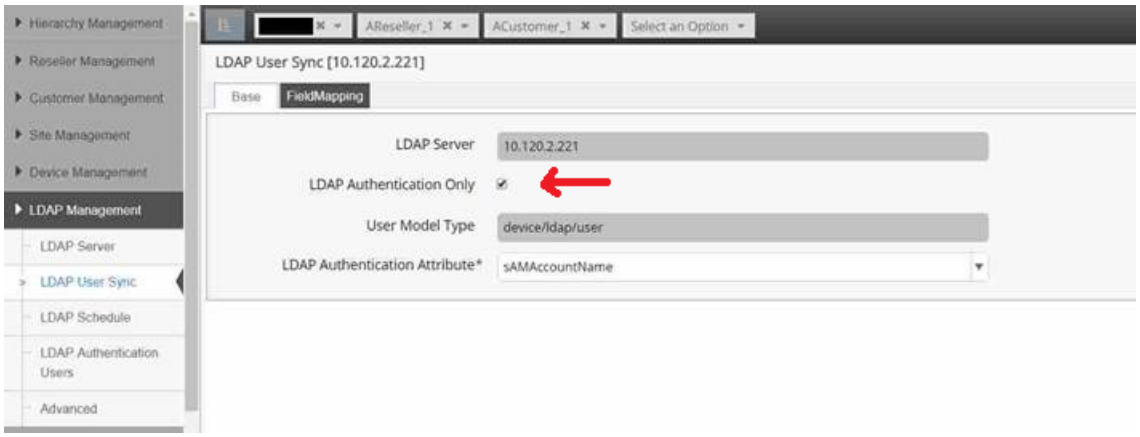


3. Enable the *Cisco DirSync* service on CUCM. Go to **Cisco Unified Serviceability Tools > Service Activation**. At the bottom of the page you will find Cisco DirSync Service. It will take some time to complete.

4. Run an LDAP sync from CUCM. Go to **System > LDAP > LDAP Directory** and select **Perform Full Sync Now**.



5. Check the user's status in CUCM. The user status should now display as "Active LDAP synchronized user"
6. In Automate, add the LDAP User Sync again and enable the LDAP Authentication Only option.

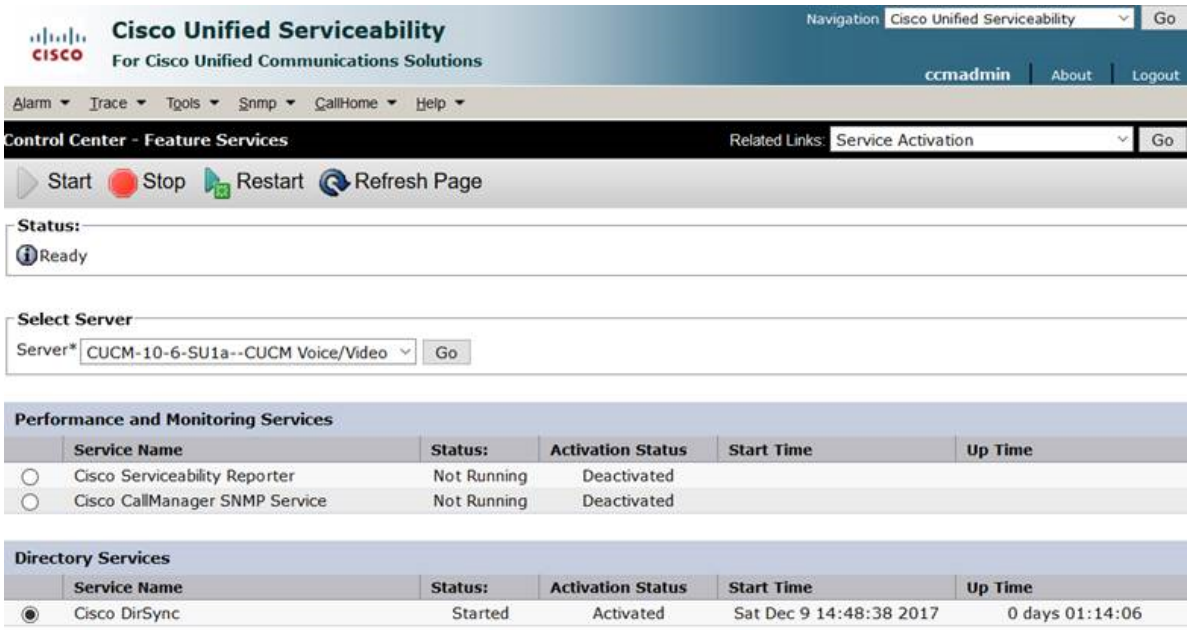


7. Run a DataSync from VOSS Automate with CUCM, that is, the data sync with name that starts with "HcsPull".

6.1.4. Change the LDAP user data sync back to top-down

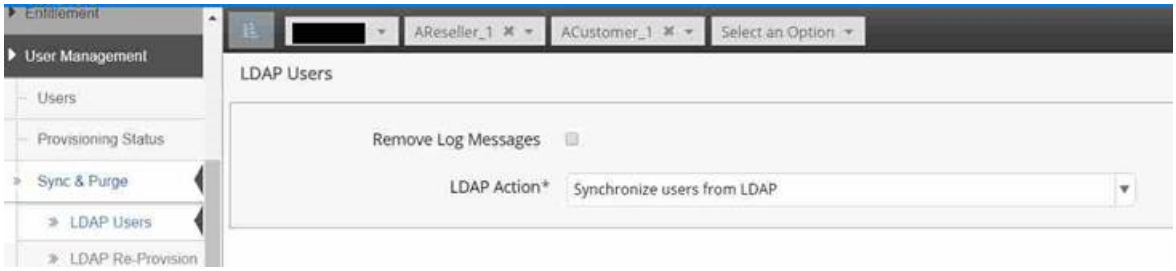
1. Stop the DirSync service on CUCM.

   Log into the CUCM Cisco Unified Serviceability page and go to **Tools > Control Center - Feature Services**. Select the Cisco DirSync service option and click **Stop**.



If this move is permanent, stop and deactivate the Cisco DirSync service on CUCM.

2. In Automate, remove the Authenticate Only LDAP User sync.
3. In Automate, add an LDAP User Sync to do full LDAP syncs. (Or you can just import the JSON file exported earlier.)
4. Go to **User Management > Sync & Purge > LDAP Users** and run the sync users from LDAP (Unselect the Remove Log Messages).



5. Check user in CUCM and in Automate. The user status should be:
   - CUCM: “LDAP Active Synced”
   - Automate: “VOSS-LDAP Synced”

7. Data Sync Workflows

7.1. Data sync workflows

7.1.1. Overview

Data sync workflows in Automate are processes that manage the synchronization of data between different systems or components.

The data sync workflows described in this topic form part of the set of workflows that are carried out when adding a new CUCM cluster. The name and description of each calling workflow is indicated, as well as the corresponding information for called or included workflows.

7.1.2. UserCucmSyncAdd

1. Include PWF: UserCucmSyncAdd_ValidateData: Validate the CUCM user before any work is done. If errors are found the CUCM user will be purged again and transaction will exit.
2. Include PWF: UserDiscoverApps : Discover all user apps (fn.user_discover_apps pwf.username)
3. Initialize minimum user input: user_input_role_selfservice, pwf.user_detail.is_cucm_user, entitlement_profile
4. Add a new Local User if it does not exist using just the username as input and role as selfservice
5. Update Local User if it exists
6. Discover all user apps again after user add
7. Include PWF UserMoveCucmUsingFilter: Move the CUCM user if it matches a filter
8. Include PWF UserUpdateApps: Update data on all apps where data is mapped using the latest user data as input
9. Include PWF UserCucmOverbuild: Move CUCM user if configured in Global Flow-Through-Provisioning settings
10. Include PWF UserCucmFlowThroughProvisioning: Flow through provision user if configured in Site Defaults Doc settings

7.1.3. UserCucmSyncUpdate

This data sync workflow updates a user from CUCM sync.

1. Include PWF UserHandleUserNameChange: Handle CUCM username change if the username changed (this also has its own call to UserDiscoverApps inside it).
2. Update user if it exists
3. If VOSS user, also known as "Local User" (`data/User`) does NOT exist, then CREATE it at the same node as `cucm/User` using just the username as input, and role as "selfservice"
    a. Include PWF: UserDiscoverApps : Discover all user apps
4. Include PWF: UserUpdateApps : Update data on all apps where data is mapped using the latest user data as input

7.1.4. UserCucmSyncRemove

This data sync workflow removes a user from CUCM sync.

1. Build a list of associated Device Objects
2. Build a list of associated Device Profile Objects
3. Include PWF UserDiscoverApps: Discover all user apps
4. Include PWF RemoveDeviceObjectFromCucmPWF: REMOVE each Phone (cucm/Phone) via [RemoveDeviceObjectFromCucmPWF]
5. Include PWF RemoveDeviceObjectFromCucmPWF: REMOVE each cucm/DeviceProfile via workflow [RemoveDeviceObjectFromCucmPWF]
6. Remove Voicemail Account (device/cuc/User) if retain Voicemail is set to false and the removed User had an associated Voicemail Account
7. Remove User only if its new sync source is LOCAL/CUC/WEBEX_TEAMS user AND retain Voicemail / Webex Teams are set to false AND User does not have other services

HcsAddUpdateCucmPhoneDsPWF

1. Include PWF HcsAddUpdateCucmPhoneForeachUserPWF: (lines_removed)
2. Include PWF LineDeletion_after_DeviceUpdate_PWF: INCLUDE sub-pwf to CLEANUP all associated cucm/Line instances , depending on Line Delete Preferences

7.1.5. LineDeletion_PhoneDataSync_PWF

This data sync workflow is a shared PWF, used typically by data-syncs, to maintain/delete cucm/Lines and INI after the sync detected that, for example, device was removed.

1. SET var [pwf.deletedDeviceDat]: if the calling/parent PWF had already defined this var, then use that pre-defined value, else default to use first [previous] or else [input] if previous is not available
2. SET pwf VARS: unique CUCM line, deviceOwner, allowLineDeletion, allowLineUpdate, lineUpdateCft, lineHn, lines_removed

3. Include PWF LineDeletionWrapperProcessLinesListPWF: If either of the LineDeletion-prefs are being used / have been set, then proceed with: ForEach line, run the line cleanup/maintainance workflow

7.1.6. LineDeletion_after_DeviceUpdate_PWF

1. To maintain/delete cucm/Lines and INI after the sync detected that e.g. device was removed
2. If either of the LineDeletion-prefs are being used / have been set, then proceed with: ForEach line, run the line cleanup/maintainance workflow:
    a. Includes LineDeletionWrapperProcessLinesListPWF - If either of the LineDeletion-prefs are being used / have been set, then proceed with this line of workflows.
        i. For each line, run the line cleanup/maintainance workflow = LineDeletion_ProcessLine_PWF (remove device/cucm/Line)
    b. Includes RemoveAgentLineRelPWF" (Remove Agent Line Relation will fire if it is not an update via Sync or if it is an Sync update and lines have been removed.)
        i. Remove relation/HcsCucmCcTagREL

7.1.7. HcsPreSyncSipGwSiptrunkUpdatePWF

This data sync workflow updates SipTrunk field in SIP Gateway if siptrunk name changes.

- Includes: HcsPreSyncUpdateSipGwPWF
    a. This workflow updates SipTrunk field in SIP Gateway if siptrunk fields (ipAddress, port) change.

7.1.8. PWF_HcsPostSyncSipGwSiptrunkDeletePWF

This data sync workflow deletes SIP GW which is using this SIP trunk.

- Includes: HcsPostSyncDeleteSipGwPWF
    a. Removes relation/*HcsSipGwREL* where name = input.sipgw.name

8. Number Inventory Customization

8.1. Number inventory flexibility and description customization

8.1.1. Overview

When number inventory changes are made while using Automate features, users are provided with a mechanism to define the logic for the dynamic population of a set of number inventory fields.

To make use of the mechanism, the following elements should be noted:

- Only a subset of number inventory fields can be managed.
- The mechanism provided includes:
  - A Configuration Template (CFT) called `IniUpdateCustomCFT`
  - A set of macros to manage the `description` field. These are comprised of a "caller" macro entered into the CFT, which in turn references a set of macros that apply in accordance with the specified feature associated with the macro.
  - If the "caller" macro is added to the CFT, a set of default values are added to the `description` field when the associated feature is used.
  - Users therefore have a choice to either modify the "caller" macro representing the feature use logic, or the individual macros called by it.

8.1.2. Managed and non-managed number inventory fields

Number Inventory data fields are of 2 types:

- Automate managed number inventory fields.

  The values of these fields are derived from workflows and cannot be customized.

  Managed number inventory fields are:

```
status
usage
e164number
vendor
internal_number_type
reservation_notes
```

- Automate non-managed number inventory fields.

  The values of these fields are updated when a number is marked used, available, etc.

  By default, when a number is marked as available, the non-managed `extra-` fields retain their value and are not cleared. Customers have options available to manage these with a provided Configuration Template (CFT) called: `IniUpdateCustomCFT`.

  For details, see the topic *Persisting and Modifying Values in Extra Fields* in the Core Feature Guide.

  Non-managed number inventory fields are all those fields that are not included in the Managed number inventory fields.

8.1.3. Number inventory description field customization

The `description` field in the number inventory is cleared when a number is marked as available. However, a "caller" macro is available that can be referenced in the provided Configuration Template: `IniUpdateCustomCFT`.

To use the CFT and macros, clone `IniUpdateCustomCFT` from `sys` to the required hierarchy and set `description` field to:

```
{{ macro.INI_Description_From_Caller_Workflow }}
```

Using this "caller" macro allows customers to determine how to populate the `description` field for different scenarios (identified by `pwf.ini_caller` values in the macro) - number assignment, unassignment, the usage context (e.g users, phone, Auto Attendant, Call queue, etc). The "caller" macro references a set of individual macros that apply for various usage scenarios. These macros can in turn be customized to allow for a required update of the number inventory.

The default `description` values of these macros, as well as the scenario and macro name are listed below:

Named macro: `INI_Description_From_Caller_Workflow` defines a macro per context (`pwf.ini_caller`):

pwf.ini_caller	Feature	Macro Name
INILineAdd	Used by Quick User and AddPhone	INI_Description_INILineAdd
UserMoveApps	Used by Quick User	INI_Description_UserMoveApps
Cisco_CallHandler	Used by CallHandler	INI_Description_Cisco_CallHandler
Cisco_CPUG	Used by CallPickupGroups	INI_Description_Cisco_CPUG
Cisco_HG	Used by HuntGroups	INI_Description_Cisco_HG
Cisco_SubMove	Used by User Move	INI_Description_Cisco_SubMove
Cisco_ChangeLine	Used by User Change Line	INI_Description_Cisco_ChangeLine
Cisco_Reassign	Used by Re-assign services	INI_Description_Cisco_Reassign
Cisco_CallPark		INI_Description_Cisco_CallPark
Cisco_CTIRP		INI_Description_Cisco_CTIRP
Cisco_MeetMe		INI_Description_Cisco_MeetMe
Microsoft_QAS		INI_Description_Microsoft_QAS
Microsoft_QAS_Hybrid_Only		INI_Description_Microsoft_QAS
Hybrid_AddSnr		INI_Description_Hybrid_AddSnr
Hybrid_AddMVS		INI_Description_Hybrid_AddMVS

Macro Name	Default Macro Description Value
INI_Description_INILineAdd	input.description, else macro.DISPLAY_NAME_FNAME_LNAME_MAX_40_CHARS
INI_Description_UserMoveApps	input.description, else macro.DISPLAY_NAME_FNAME_LNAME_MAX_40_CHARS
INI_Description_Cisco_CallHandler	input.DisplayName
INI_Description_Cisco_CPUG	input.description
INI_Description_Cisco_HG	input.description
INI_Description_Cisco_SubMove	pwf.description_Display_Name
INI_Description_Cisco_ChangeLine	data.User.display_name | username:input.username
INI_Description_Cisco_Reassign	macro.DISPLAY_NAME_FNAME_LNAME_MAX_40_CHARS
INI_Description_Cisco_CallPark	pwf.description
INI_Description_Cisco_CTIRP	input.description
INI_Description_Cisco_MeetMe	input.description
INI_Description_Microsoft_QAS	if pwf.workflow_source = MS_TEAMS input.FirstName and input.LastName else input.first_name input.last_name
INI_Description_Hybrid_AddSnr	input.first_name input.last_name
INI_Description_Hybrid_AddMVSa	pwf.dataUserObject.first_name pwf.dataUserObject.last_name

8.1.4. Macro details

Caller macro

Clone the caller macro to your hierarchy and modify if needed. The called macro references associated with a feature can be substituted with custom macros.

- **name**: `INI_Description_From_Caller_Workflow`
- **description**: "Calculation of INI description field using pwf.ini_caller value to identify required macro. For use by INIUpdateCustomCFT"

macro (Line breaks added for readability):

```
"macro": "(( pwf.ini_caller == INILineAdd )) <{{ macro.INI_Description_INILineAdd }}>
          (( pwf.ini_caller == UserMoveApps )) <{{ macro.INI_Description_UserMoveApps }}>
          (( pwf.ini_caller == Cisco_CallHandler )) <{{ macro.INI_Description_Cisco_
↪CallHandler }}>
          (( pwf.ini_caller == Cisco_CPUG )) <{{ macro.INI_Description_Cisco_CPUG }}>
          (( pwf.ini_caller == Cisco_HG )) <{{ macro.INI_Description_Cisco_HG }}>
          (( pwf.ini_caller == Cisco_SubMove )) <{{ macro.INI_Description_Cisco_SubMove }
↪}>
          (( pwf.ini_caller == Cisco_ChangeLine )) <{{ macro.INI_Description_Cisco_
↪ChangeLine }}>
          (( pwf.ini_caller == Cisco_Reassign )) <{{ macro.INI_Description_Cisco_
↪Reassign }}>
          (( pwf.ini_caller == Cisco_CallPark )) <{{ macro.INI_Description_Cisco_
↪CallPark }}>
          (( pwf.ini_caller == Cisco_CTIRP )) <{{ macro.INI_Description_Cisco_CTIRP }}>
          (( pwf.ini_caller == Cisco_MeetMe )) <{{ macro.INI_Description_Cisco_MeetMe }}>
          (( pwf.ini_caller == Microsoft_QAS )) <{{ macro.INI_Description_Microsoft_QAS }
↪}>
          (( pwf.ini_caller == Microsoft_QAS_Hybrid_Only )) <{{ macro.INI_Description_
↪Microsoft_QAS }}>
          (( pwf.ini_caller == Hybrid_AddSnr )) <{{ macro.INI_Description_Hybrid_AddSnr }
↪}>
          (( pwf.ini_caller == Hybrid_AddMVS )) <{{ macro.INI_Description_Hybrid_AddMVS }
↪}>
          <{{ fn.drop }}>"
```

Called macros

Clone the called macro to your hierarchy and modify if needed. The custom macro is then called for the associated feature by the caller macro.

For macro details, refer to the Named Macro Reference HTML document for your release on the documentation portal.

9. Custom Configuration

9.1. Configuration overview

Administrators with sufficient privileges have access to a number of resources that can be configured for various purposes.

Configuration can include:

- clone and modification of Configuration Templates
- the use of named macros in Configuration templates or resources
- Modification of the Site Defaults Doc
- Modification of the Quick Add Group

By default, Automate provides a number of defaults in a collection of objects. Some of these defaults, such as the Default Site Dial Plan, are created when a Customer hierarchy is created. When a Site hierarchy is created, the defaults act as templates to provide site-specific values for the use of for example user creation at site level.

In particular, the following default templates result in site specific instances when a site is created:

- Default Site Dial Plan
- Default Site Defaults Doc
- Default Quick Add Group

Administrators who have access to these resources can customize these templates so that all subsequent sites that are created, have modified site specific instances.

At a site level, the instances themselves can be modified so that users that are added at the site, have these custom values applied to them. Site level modification can thus be made to:

- Site level Site Defaults Doc
  - Contains site-specific values
  - Values are referenced by Configuration Templates at site level
  - Some values referenced by the Quick Add Group, for example Default CUC User Template
- Site level Quick Add Group
  - Contains Configuration templates used when adding users (Quick Add User)

Customization of resources may involve:

- changing values (optionally using macros)

- changing or replacing Configuration Templates with newly cloned and created ones (optionally using macros)

9.2. Site defaults reference

When the end-to-end feature is installed and the system is provisioned for users, reference data representing a sample dial plan is stored as a template instance of a Site Defaults data model.

This reference data shows text values or combines default text and references with the feature's macros and configuration templates.

For example, when site hierarchy and user administration is carried out, input administrator data (stored as for example a named macro called SITENAME) is combined with this reference data to provide default or generated values. If a customer administrator at for customer "VS-Corp" created a site called "Boston", the line and call partitions associated with the site are called "Site-Boston" when the macro is resolved.

The sample dial plan is an instance of the site defaults and contains illustrative values. Where the name of the data refers to a Template, the Configuration Template itself contains or results in further data defaults that are applied when the End-to-End feature is used for user and site administration.

For example, the template called "Default CUCM RDP Template" sets a number of default Remote Destination Profile values when it is added:

Name	Value
product	"Remote Destination Profile"
protocol	"Remote Destination"
description	"Created by default template"
callInfoPrivacyStatus	"Default"
protocolSide	"User"
devicePoolName	"{{macro.CUCM_PHONE_devicePoolName}}"
class	"Remote Destination Profile"

The specified values for the CUCM Remote Destination Profile data are defaulted, while the macro called *macro.CUCM_PHONE_devicePoolName* looks up the Default Device Pool value for the Dial Plan name obtained from the Site Defaults instance. This value is based on the Customer ID and Site ID, for example "Cu12Si3-DevicePool" where the Customer ID is 12 and the Site ID is 3. This value is based on an ID lookup. These IDs are not reset during subsequent imports of the sample dial plan, so that existing instances on the system that use an ID will not be overwritten.

Most of these settings are populated with default values based on an example dial plan that ships with the core product. Typically, HCS templates overwrites these settings with different default values which are in line with a required dial plan.

All of the Day-2 features (provisioning of user, phone, line, and so on) have macros that point to the Site Defaults Document (SDD) when viewing the features in the GUI. This serves two main purposes:

1. It simplifies the administration of these features, because as soon as the administrator opens a GUI form to provision any of these features, the bulk of the settings are pre-populated with correct values (based on correct values in the SDD).
2. A abstracted view of Day-2 templates is possible by means of Field Display Policies that can hide complex and technical information while still populating these (hidden from view) by means of the SDD.

If changes are made in the SDD, for example new attributes are added, then pre-existing SDD instances must be manually updated to include these new attributes (if necessary).

9.3. Introduction to named macros

Named macros relate to two resources:

- Site Defaults Doc

  These macros have a syntax is as in the following example:

  `data.SiteDefaultsDoc.defaultcuchtmlnotificationtemplate`

  Here the macro references an attribute of the specific Site Defaults Doc that is available at the hierarchy that the macro is called. In this case, the attribute name that is referenced, is `defaultcuchtmlnotificationtemplate`.

  The Site Defaults Doc template that is used to resolve to values in site specific Site Defaults Doc instances is only configurable by an administrator with the *global* administrator role. Other provider, and customer level administrators can modify the individual instances at the specific site, while administrators at a site level can only inspect the values.

- Customizable Configuration Templates

  Administrators - typically at provider or customer level have a list of Configuration Templates available from the **System Configuration** menu. These can be modified and cloned to a required hierarchy if customization is needed. In these Configuration Templates, macros may be referenced.

  If the value to which a named macro resolves is required in a custom Configuration template, the named macro can then be added to the Configuration Template. Note however that Configuration Template customization involves more than the use of named macros.

The lists of named macros provides the macro name and its source code as reference. The macro name and the macro code which references to the Site Defaults Doc model attribute name describe its value in the Site Defaults Doc. No further explanation is needed in this case.

Where needed and the macro code is not self explanatory, a comment is added at the macro. Line breaks have been added to long macro source code strings.

The example below shows naming conventions, descriptions and line breaks.

- **macro.CUC_HTML_NotificationTemplateName**

```
{{ data.SiteDefaultsDoc.defaultcuchtmlnotificationtemplate
   | name:macro.DPSITE }}
```

  By default, the macro: `DPSITE` resolves to the site name at the current hierarchy, in other words: `{{ data.BaseSiteDAT.SiteName }}`

  The macro therefore resolves to the CUC notification template with the same name as the current site.

9.4. Customizing quick add groups

Quick Add Groups provide a selection of service defaults for use in user management, particularly when using Quick Add User.

One or more Quick Add Groups may be available at a site. You'll choose a Quick Add Group during the Quick Add User workflow.

Administrators at customer hierarchy level and higher can modify Quick Add Groups. By default, a Quick Add Group is created for each site when it is created. Service defaults that are applied at the site level are stored in a read-only Quick Add Group called "default", and serve as a template for the one created at the site. Service defaults available in a Quick Add Group are stored as configuration templates (CFTs) to services.

The following types of customization is done at site level:

- Add new Quick Add Groups (and CFTs)
- Add and select different CFTs to be used in an existing Quick Add Group

Related topics

- Configuration Templates in the Core Feature Guide.
- Manage Sites in the Core Feature Guide.
- Quick Add Groups in the Core Feature Guide.
- Quick Add User in the Core Feature Guide.
- Quick User (Webex App) in the Core Feature Guide.
- Quick Add User (Microsoft) in the Core Feature Guide.

9.5. Quick Add User Configuration

9.5.1. Quick add user provisioning workflow structure

The Quick Add User functionality uses a `view` model to define its schema that also ultimately defines:

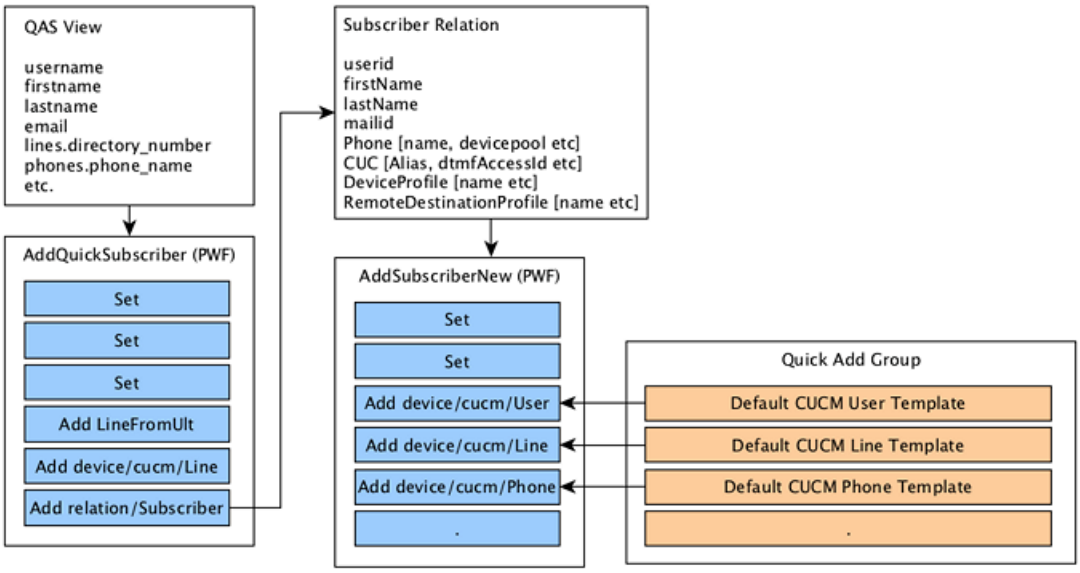
- The form fields a user sees in the GUI
- The column headings in the Quick Add User bulk loader sheet
- The attributes supported by the API endpoint (`view/QuickSubscriber`)

This `view` schema represents a simplified version of all possible user fields.

Associated with the view is a Provisioning Workflow (PWF) called `AddQuickSubscriber`, which captures these simplified fields and combines or maps them into the various components (services) of a full user. The main purpose of this workflow is to execute an "Add `relation/Subscriber`" operation.

The "Add `relation/Subscriber`" operation in turn executes a *second* PWF called `AddSubscriberNew`, which separates out the data for each user component or service and executes an "Add" operation on the corresponding device models, for example on `device/cucm/Phone`.

It is inside this `AddSubscriberNew` workflow that the configuration templates of the selected Quick Add Group is evaluated.



Since the Quick Add Group configuration templates (CFTs) are evaluated inside the `AddSubscriberNew` workflow, it is important that the user `relation` field names *not* be those of the Quick Add User view. Generally, all field values are made available to an executing PWF and is referred to as the *context*.

For example, refer to the diagram and notice the difference in field (attribute) naming between the Quick Add User view and the user `relation` schemas, for example `username` as opposed to `userid`. The difference is subtle, but very important: the user name in the Quick Add User workflow context (`AddQuickSubscriber`) is `username`, but in the user `relation` workflow context (`AddSubscriberNew`) it is `userid`.

9.5.2. User relation context

When adding a user, multiple services are often added at the same time. For example, multiple phones with multiple lines.

During the "Add" operation, the `AddSubscriberNew` workflow therefore iterates (loops) over these added services and makes the data for each instance available in a *context variable*.

Consider for example the following workflow input context for adding a simple user:

```
{
    "input": {
        "userid": "TestUserX77",
        "firstName": "Test",
        "lastName": "UserX77",
        "mailid": "testuserx77@mail.com",
        "Phone": [
            {
                "name": "SEP111222333444",
                "product": "Cisco Unified Client Services Framework",
                "protocol": "SIP",
                "lines": {
                    "line": [
                        {
```

(continues on next page)

(continued from previous page)

```
                            "dirn": {
                                "pattern": "82007",
                                "routePartitionName": "Cu1-DirNum-PT"
                            }
                        }
                    ]
                }
            },
            {
                "name": "SEP000222333444",
                "product": "Cisco Unified Client Services Framework",
                "protocol": "SIP",
                "lines": {
                    "line": [
                        {
                            "dirn": {
                                "pattern": "82008",
                                "routePartitionName": "Cu1-DirNum-PT"
                            }
                        }
                    ]
                }
            }
        ]
    }
}
```

The input context shows there are two phones, each with one line. The `AddSubscriberNew` workflow loops over the phones and put the entire contents of each instance in a context macro called `{{ input.PhoneX }}`.

Therefore, when it iterates over the first phone, `{{ input.PhoneX }}` will be equal to:

```
{
    "name": "SEP111222333444",
    "product": "Cisco Unified Client Services Framework",
    "protocol": "SIP",
    "lines": {
        "line": [
            {
                "dirn": {
                    "pattern": "82007",
                    "routePartitionName": "Cu1-DirNum-PT"
                }
            }
        ]
    }
}
```

In particular, the input context variable `{{ input.PhoneX.name }}` will be equal to `SEP111222333444`. During the second iteration, `{{ input.PhoneX.name }}` will be equal to `SEP000222333444`.

9.5.3. Config template looping and merging overview

When using Quick Add User functionality, the user *relation* workflow (`AddSubscriberNew`) follows the same pattern for each device model being added.

A base *config template* for the model is merged with the template specified in the Quick Add Group.

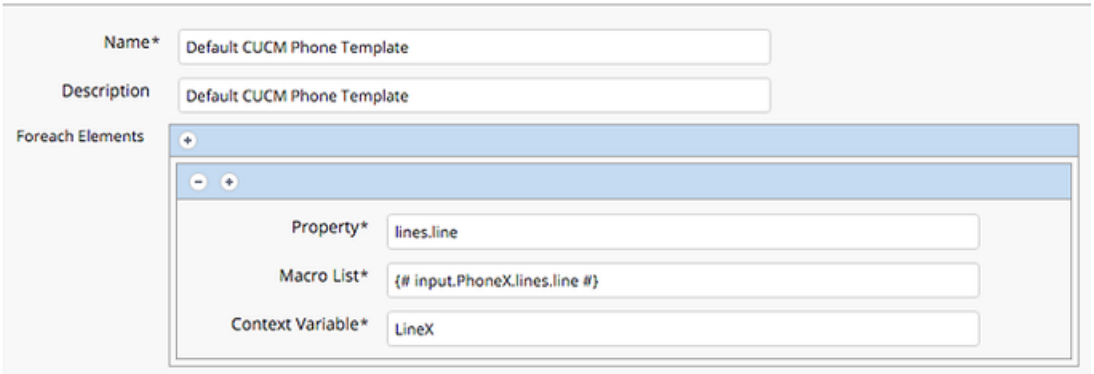
The base template sets fields that need to be hard coded to fulfill certain business rules, for example forcing the *ownerUserName* on phone to the user's *userid*.

This merging of config templates works seamlessly for the most part. However, the merge process becomes a little more complex when the config templates being merged contain lists that need to be looped through.

9.5.4. Config template looping

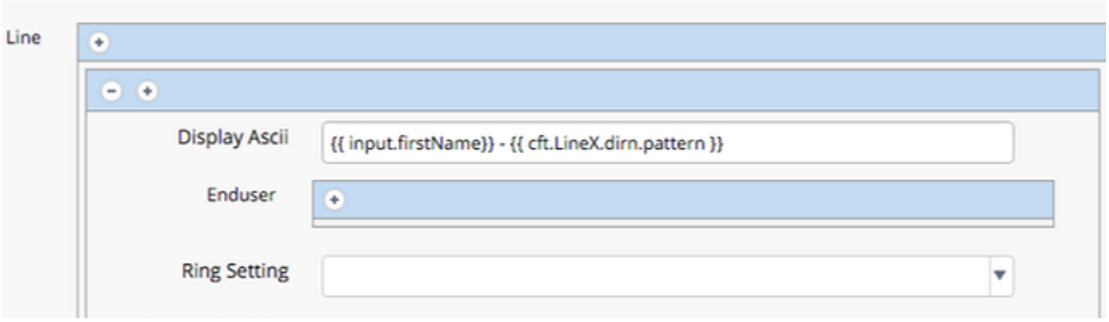
An example of a Config Template (CFT) loops is found in the Phone CFT because the Phone device model contains a list of Lines. In order to set template values for some of these line fields, one needs to iterate over the lines inside the CFT.

CFTs support this looping (foreach) mechanism, as can be seen on the GUI form when looking at a specific CFT instance.



The above screenshot illustrates how such a loop (Foreach) can be configured. The CFT is configured to loop over the macro `{# input.PhoneX.lines.line #}`. Remember, the `PhoneX` context variable contains a single instance of Phone data. So by referencing `lines.line`, we are referencing the *list of lines* associated to this specific Phone.

The Property indicates which field in the CFT we want to extend with each element of the `lines.line` list. Finally, the contents of each `lines.line` instance will be stored in `{{ cft.LineX }}` with each iteration. The `cft` namespace indicates the origin of the variable `LineX`.



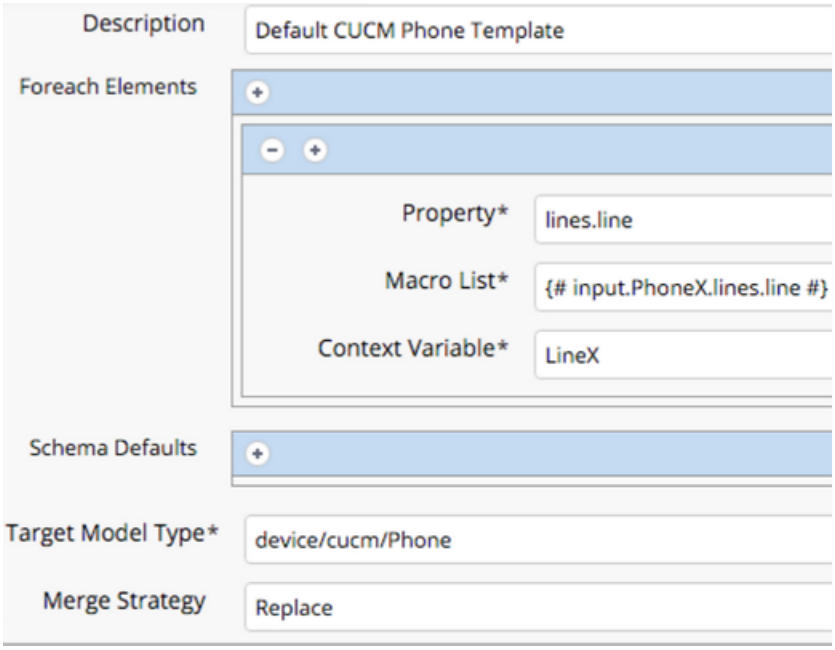
The screenshot above illustrates how the iterated variable can be used in the `lines.line` field of the CFT.

The first macro in the Display Ascii input box will reference the `firstName` of the user in the user context, while the second macro references the pattern of the line instance being iterated over.

9.5.5. Config template merge strategy

Given the example shown in the Config Template Looping topic, the use of CFTs in workflows is slightly more complicated, because the base CFT might already be implementing a CFT loop as seen in the example.

In this case, it is required for the QAG CFT to specify the correct Merge Strategy.



As can be seen from the image, since the base Phone CFT in the Subscriber Relation already implements a loop over lines, it is necessary for the QAG template to specify a “Replace” instead of “Additive” Merge Strategy.

This “replace” merge strategy allows the resulting lists of lines from the evaluated CFTs to be merged instead of being added together.

For reference, the following base CFTs implement a foreach loop:

- Phones (`AddSubscriber_PhoneLoop`)
  - `lines.line`
  - `lines.line.0.associatedEndusers.enduser`
  - `services.service`
- Device Profile (`AddSubscriber_DPLoop`)
  - `lines.line`
  - `lines.line.0.associatedEndusers.enduser`
  - `Services.service`
- Remote Destination for Dual Mode Phone (`AddSubscriber_RDLoopPhone`)

  - `lineAssociations.lineAssociation`
- Remote Destination Profile SNR (`AddSubscriber_RDPLoop`)
  - `lines.line`
  - `lines.line.0.associatedEndusers.enduser`
- Remote Destination SNR (`AddSubscriber_RDLoop`)
  - `lineAssociations.lineAssociation`

9.6. Smart Add Phone Configuration

9.6.1. Custom line settings for Smart Add Phone configuration template

In order to customize the line settings of a custom Configuration Template to be used specifically with the Smart Add Phone feature, the cloned, customized template should be exported as a JSON file and edited manually.

Consider the `data` attribute snippet of an example Phone template JSON file *before* customization for custom lines:

```
"data": {
    "description": "Custom CUCM Phone Template",
    "name": "Custom Cisco 7945",
    "target_model_type": "device/cucm/Phone",
    "template": {
        "protocol": "SCCP",
        "softkeyTemplateName": "Standard Agent",
        "phoneTemplateName": "Standard 7945 SCCP",
        "callingSearchSpaceName": "(( input.Phone.callingSearchSpaceName == None )) \
          <{{macro.CUCM_PHONE_callingSearchSpaceName}}> \
          <{{input.Phone.callingSearchSpaceName}}>",
        "servicesUrl": "Both",
        "builtInBridgeStatus": "On",
        "useTrustedRelayPoint": "Default",
        "userlocale": "English United States",
        "enableExtensionMobility": "(( True ))",
        "commonDeviceConfigName": "Agent_CDC",
        "networkLocale": "United States",
        "packetCaptureMode": "None",
        "product": "Cisco 7945",
        "description": "Created by Custom Phone Template",
        "userLocale": "English United States",
        "deviceMobilityMode": "On",
        "certificateOperation": "No Pending Operation",
        "class": "Phone",
        "securityProfileName": "Cisco 7945 - Standard SCCP Non-Secure Profile",
        "protocolSide": "User",
        "commonPhoneConfigName": "Standard Common Phone Profile"
    ...
```

To customize this template for use with the Smart Add Phone feature and to add custom line settings, it requires additional elements in the `data` attribute group:

- A `foreach` loop method over lines with a line variable, e.g. `LineX`:

```
"foreach": [
    {
        "property": "lines.line",
        "context_var": "LineX",
        "macro_list": "{# input.lines #}"
    }
],
```

- Line specific customization for each line added on the Smart Add Phone input form on the Admin Portal represented by an attribute `lines` that contains a `line` list.

  An example is shown below:

```
"lines": {
     "line": [
         {
             "maxNumCalls": "2",
             "displayAscii": "Test Site Phone",
             "busyTrigger": "1",
             "label": "{{ cft.LineX.directory_number }}",
             "e164Mask": "8005551212",
             "callInfoDisplay": {
                 "dialedNumber": "(( True ))",
                 "callerName": "(( True ))"
             },
             "asciiLabel": "{{ cft.LineX.directory_number }}",
             "display": "Test Site Phone"
         }
     ]
 }
```

- Added attribute to template called `merge_strategy` and set to `replace`.

The example `data` attribute group below shows the additional customization integrated into the template. This template can then be selected from the **Phone Template** dropdown to customize the phone's line settings:

```
"data": {
    "name": "Custom Cisco 7945",
    "description": "Custom Smart Add Phone Template - Cisco 7945",
    "foreach": [
        {
            "property": "lines.line",
            "context_var": "LineX",
            "macro_list": "{# input.lines #}"
        }
    ],
    "merge_strategy": "replace",
    "target_model_type": "device/cucm/Phone",
```

(continues on next page)

(continued from previous page)

```
    "template": {
        "protocol": "SCCP",
        "softkeyTemplateName": "Standard Agent",
        "phoneTemplateName": "Standard 7945 SCCP",
        "callingSearchSpaceName": "(( input.Phone.callingSearchSpaceName == None )) \
          <{{macro.CUCM_PHONE_callingSearchSpaceName}}> \
          <{{input.Phone.callingSearchSpaceName}}>",
        "servicesUrl": "Both",
        "builtInBridgeStatus": "On",
        "useTrustedRelayPoint": "Default",
        "userlocale": "English United States",
        "enableExtensionMobility": "(( True ))",
        "commonDeviceConfigName": "Agent_CDC",
        "networkLocale": "United States",
        "packetCaptureMode": "None",
        "product": "Cisco 7945",
        "description": "Created by Custom Phone Template",
        "userLocale": "English United States",
        "deviceMobilityMode": "On",
        "certificateOperation": "No Pending Operation",
        "class": "Phone",
        "securityProfileName": "Cisco 7945 - Standard SCCP Non-Secure Profile",
        "protocolSide": "User",
        "commonPhoneConfigName": "Standard Common Phone Profile",
        "lines": {
            "line": [
                {
                    "maxNumCalls": "2",
                    "displayAscii": "Test Site Phone",
                    "busyTrigger": "1",
                    "label": "{{ cft.LineX.directory_number }}",
                    "e164Mask": "8005551212",
                    "callInfoDisplay": {
                        "dialedNumber": "(( True ))",
                        "callerName": "(( True ))"
                    },
                    "asciiLabel": "{{ cft.LineX.directory_number }}",
                    "display": "Test Site Phone"
                }
            ]
        },
        ...
```

10. Configuration Reference

10.1. Reference material

This section covers reference material for use in the customization of the Quick Add User feature in particular and custom configuration in general.

The material is divided into:

- Site Defaults Reference
- Quick Add Groups Configuration Template Reference
- Named Macro Reference
- Quick Add User Workflow Macro Reference

10.2. Site Defaults Macros

10.2.1. Lines - site defaults

Configuration template applied to: `line-cft`

- Default CUCM Line Partition
  - **Macro**: CUCM_LINE_routePartitionName
  - **Default Value**: Site-{{macro.SITENAME}}
- Default CUCM Line CSS
  - **Macro**: CUCM_LINE_shareLineAppearanceCssName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward No Answer CSS
  - **Macro**: CUCM_LINE_callForwardNoAnswer_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward Busy Internal CSS
  - **Macro**: CUCM_LINE_callForwardBusyInt_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward No Answer Internal CSS

  - **Macro**: CUCM_LINE_callForwardNoAnswerInt_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward No Coverage CSS
  - **Macro**: CUCM_LINE_callForwardNoCoverage_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward No Coverage Internal CSS
  - **Macro**: CUCM_LINE_callForwardNoCoverageInt_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward On Failure CSS
  - **Macro**: CUCM_LINE_callForwardOnFailure_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward On Failure Internal CSS
  - **Macro**: CUCM_LINE_callForwardNotRegisteredInt_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward Not Registered CSS
  - **Macro**: CUCM_LINE_callForwardNotRegistered_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward Busy CSS
  - **Macro**: CUCM_LINE_callForwardBusy_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward Alternate Party CSS
  - **Macro**: CUCM_LINE_callForwardAlternateParty_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward Secondary CSS
  - **Macro**: CUCM_LINE_callForwardAll_secondaryCallingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Line Call Forward All CSS
  - **Macro**: CUCM_LINE_callForwardAll_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}

10.2.2. Phones - site defaults

Configuration template applied to: `SubscriberPhonePrePopulate`

- Default CUCM Line Partition
  - **Macro** CUCM_LINE_routePartitionName
  - **Default Value** Site-{{macro.SITENAME}}
- Default CUCM Phone Product
  - **Macro**: CUCM_PHONE_product
  - **Default Value**: Cisco 9971
- Default CUCM Phone Security Profile
  - **Macro**: CUCM_PHONE_securityProfileName
  - **Default Value**: Cisco 9971 - Standard SIP Non-Secure Profile
- Default CUCM Phone Protocol
  - **Macro**: CUCM_PHONE_protocol
  - **Default Value**: SIP
- Default CUCM Phone Common Profile
  - **Macro**: CUCM_PHONE_commonDeviceConfigName
  - **Default Value**: Standard Common Phone Profile
- Default CUCM Location
  - **Macro**: CUCM_PHONE_locationName
  - **Default Value**: Cu{{data.BaseCustomerDAT.InternalCustomerID}} Si{{data.BaseSiteDAT.InternalSiteID}}-Location
- Default CUCM Phone Button Template
  - **Macro**: CUCM_PHONE_phoneTemplateName
  - **Default Value**: Standard 9971 SIP
- Default CUCM Device Pool
  - **Macro**: CUCM_PHONE_devicePoolName
  - **Default Value**: Cu{{data.BaseCustomerDAT.InternalCustomerID}} Si{{data.BaseSiteDAT.InternalSiteID}}-DevicePool
- Default CUCM Phone Presence Group
  - **Macro**: CUCM_PHONE_presenceGroupName
  - **Default Value**: Standard Presence group
- Default CUCM Device CSS
  - **Macro**: CUCM_PHONE_callingSearchSpaceName
  - **Default Value**: EmergencyOnly-{{macro.SITENAME}}

10.2.3. User - site defaults

Configuration template applied to: <Not Applicable>

- Default CUCM Line Partition
  - **Macro**: CUCM_LINE_routePartitionName
  - **Default Value**: Site-{{macro.SITENAME}}
- Default CUCM Phone Product
  - **Macro**: CUCM_PHONE_product
  - **Default Value**: Cisco 9971
- Default CUCM Phone Common Profile
  - **Macro**: CUCM_PHONE_commonDeviceConfigName
  - **Default Value**: Standard Common Phone Profile
- Default CUCM Location
  - **Macro**: CUCM_PHONE_locationName
  - **Default Value**: Cu{{data.BaseCustomerDAT.InternalCustomerID}} Si{{data.BaseSiteDAT.InternalSiteID}}-Location
- Default CUCM Device Pool
  - **Macro**: CUCM_PHONE_devicePoolName
  - **Default Value**: Cu{{data.BaseCustomerDAT.InternalCustomerID}} Si{{data.BaseSiteDAT.InternalSiteID}}-DevicePool
- Default CUCM Device CSS
  - **Macro**: CUCM_PHONE_callingSearchSpaceName
  - **Default Value**: EmergencyOnly-{{macro.SITENAME}}
- Default CUCM Phone Line E164 Mask
  - **Macro**: CUCM_PHONE_lines_line_e164Mask
  - **Default Value**: 021575XXXX
- Default CUCM Phone Subscribe CSS
  - **Macro**: CUCM_PHONE_subscribeCallingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Device Profile Line E164 Mask
  - **Macro**: CUCM_DP_lines_line_e164Mask
  - **Default Value**: 021575XXXX
- Default CUCM Device Profile EMCC CSS
  - **Macro**: CUCM_DP_emccCallingSearchSpace
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Device Profile Product
  - **Macro**: CUCM_DP_product

  - **Default Value**: Cisco 9971
- Default CUCM User Presence Group
  - **Macro**: CUCM_USER_presenceGroupName
  - **Default Value**: Standard Presence group
- Default CUCM User Subscribe CSS
  - **Macro**: CUCM_USER_subscribeCallingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Remote Destination DND Option
  - **Macro**: CUCM_RDP_dndOption
  - **Default Value**: Call Reject
- Default CUCM Remote Destination Profile CSS
  - **Macro**: CUCM_RDP_callingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUCM Remote Destination Profile Line E164 Mask
  - **Macro**: CUCM_RDP_lines_line_e164Mask
  - **Default Value**: 021575XXXX
- Default CUCM Remote Destination Profile ReRouting CSS
  - **Macro**: CUCM_RDP_rerouteCallingSearchSpaceName
  - **Default Value**: Intl24HrsEnh-{{macro.SITENAME}}
- Default CUC User Template
  - **Macro**: CUC_USER_templateAlias
  - **Default Value**: voicemailusertemplate

10.2.4. Quick user (subscriber) - site defaults

Config template applied to: <Not Applicable>

- Default CUCM User Template
  - **Macro**: DEFAULT_CUCM_USER_TEMPLATE
  - **Default Value**: Default CUCM User Template
- Default CUCM Phone Template
  - **Macro**: DEFAULT_CUCM_PHONE_TEMPLATE
  - **Default Value**: Default CUCM Phone Template
- Default CUCM Line Template
  - **Macro**: DEFAULT_CUCM_LINE_TEMPLATE
  - **Default Value**: Default CUCM Line Template
- Default CUCM Device Profile Template

  - **Macro**: DEFAULT_CUCM_DEVICEPROFILE_TEMPLATE
  - **Default Value**: Default CUCM Extension Mobility Template
- Default CUCM RDP Template
  - **Macro**: DEFAULT_CUCM_RDP_TEMPLATE
  - **Default Value**: Default CUCM Remote Destination Profile Template
- Default CUCM RD Template
  - **Macro**: DEFAULT_CUCM_RD_TEMPLATE
  - **Default Value**: Default CUCM Remote Destination Template
- Default CUCM Jabber iPhone Template
  - **Macro**: DEFAULT_CUCM_JABBER_IPHONE_TEMPLATE
  - **Default Value**: Default CUCM Jabber iPhone Template
- Default CUCM Jabber Android Template
  - **Macro**: DEFAULT_CUCM_JABBER_ANDROID_TEMPLATE
  - **Default Value**: Default CUCM Jabber Android Template
- Default CUC User Template
  - **Macro**: DEFAULT_WEBEX_USER_TEMPLATE
  - **Default Value**: Default CUC User Template
- Default Webex User Template
  - **Macro**: DEFAULT_CUC_USER_TEMPLATE
  - **Default Value**: Default Webex User Template

10.2.5. Voicemail - site defaults

Configuration template applied to: <Not Applicable>

- Default CUC Phone System
  - **Macro**: CUC_PHONE_SYSTEM
  - **Default Value**: PhoneSystem
- Default CUC SMPP Provider
  - **Macro**: CUC_SMPP_PROVIDER
  - **Default Value**:
- Default CUC User Template
  - **Macro**: CUC_HTML_NotificationTemplateName
  - **Default Value**: Default_Dynamic_Icons
- Default CUC HTML Notification Template
  - **Macro**: CUC_USER_templateAlias
  - **Default Value**: voicemailusertemplate

10.2.6. Webex - site defaults

Configuration template Applied: `WebExUserPrePopulate`

Menu	Macros	Default Value
<Not Applicable>	<Not Applicable>	<Not Applicable>

10.2.7. Hot dial PLAR - site defaults

Macros	Default Value	Derived From SDD Field
<Not Applicable>	HotdialTZ	Africa/Johannesburg

10.2.8. Hunt groups - site defaults

Configuration Template applied to: `HuntGroupsPrePopulate`

- Default UCM Line Call Forward Busy CSS
  - **Macro**: Intl24HrsEnh-{{macro.SITENAME}}
  - **Default Value**: CUCM_LINE_callForwardBusy_callingSearchSpaceNam
- Default UCM Line Call Forward No Answer CSS
  - **Macro**: Intl24HrsEnh-{{macro.SITENAME}}
  - **Default Value**: CUCM_LINE_callForwardNoAnswer_callingSearchSpaceNam

10.2.9. Call pickup groups - site defaults

Configuration template applied to: `CPGPrePopulate`

- Default UCM call pickup partition
  - **Macro**: CUCM_CPUG_routePartitionName
  - **Default Value**: Site-{{macro.SITENAME}}

10.3. Quick Add Groups Configuration Template Reference

10.3.1. Configuration template reference list

- Refer to the Macro Reference chapter for macro syntax and macro functions.
- Refer to the Named Macro Reference documentation for named macro details.

Default CUC User Password Template

applies to `device/cuc/UserPassword`

Default CFT values:

- CredMustChange: {{ fn.false }}

Default CUC User PIN Template

applies to `device/cuc/UserPin`

Default CFT values:

- CredMustChange: {{ fn.false }}

Default CUC User Template

applies to `device/cuc/User`

Default CFT values:

- templateAlias: {{ data.SiteDefaultsDoc.defaultcucsubscribertemplate | | direction:local }}

Default CUCM Device Profile Template

applies to `device/cucm/DeviceProfile`

Default CFT values:

- description: Created by default template
- phoneTemplateName: Standard 9971 SIP
- product: Cisco 9971
- protocol: SIP

Default CUCM Jabber Android Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: Created by default template
- deviceMobilityMode: Default
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- locationName: {{ macro.CUCM_PHONE_locationName }}

- packetCaptureMode: None
- phoneTemplateName: Standard Dual Mode for Android
- product: Cisco Dual Mode for Android
- protocol: SIP
- protocolSide: User
- securityProfileName: Cisco Dual Mode for Android - Standard SIP Non-Secure Profile
- sipProfileName: Standard SIP Profile
- useTrustedRelayPoint: Default

Default CUCM Jabber CSF Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: Created by default template
- deviceMobilityMode: Default
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- locationName: {{ macro.CUCM_PHONE_locationName }}
- packetCaptureMode: None
- phoneTemplateName: Standard Client Services Framework
- product: Cisco Unified Client Services Framework
- protocol: SIP
- protocolSide: User
- securityProfileName: Cisco Unified Client Services Framework - Standard SIP Non-Secure Profile
- sipProfileName: Standard SIP Profile
- useTrustedRelayPoint: Default

Default CUCM Jabber iPad Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: Created by default template
- deviceMobilityMode: On
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- locationName: {{ macro.CUCM_PHONE_locationName }}
- packetCaptureMode: None
- phoneTemplateName: Standard Jabber for Tablet
- product: Cisco Jabber for Tablet
- protocol: SIP
- protocolSide: User
- securityProfileName: Cisco Jabber for Tablet - Standard SIP Non-Secure Profile
- useTrustedRelayPoint: Default

Default CUCM Jabber iPhone Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: Created by default template
- deviceMobilityMode: On
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- locationName: {{ macro.CUCM_PHONE_locationName }}
- packetCaptureMode: None
- phoneTemplateName: Standard Dual Mode for iPhone

- product: Cisco Dual Mode for iPhone
- protocol: SIP
- protocolSide: User
- securityProfileName: Cisco Dual Mode for iPhone - Standard SIP Non-Secure Profile
- useTrustedRelayPoint: Default

Default Carrier Integrated Mobile Template

applies to `device/cucm/Phone`

Default CFT values:

- alwaysUsePrimeLine: Default
- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: {{ input.firstName }} {{ input.lastName }} Carrier Integrated Mobile
- deviceMobilityMode: Default
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- deviceTrustMode: Not Trusted
- dndOption: Call Reject
- locationName: {{ macro.CUCM_PHONE_locationName }}
- outboundCallRollover: No Rollover
- packetCaptureMode: None
- phoneTemplateName: Standard Carrier-integrated Mobile
- presenceGroupName: Standard Presence group
- product: Carrier-integrated Mobile
- protocol: SIP
- protocolSide: User
- securityProfileName: Carrier-integrated Mobile - Standard SIP Non-Secure Profile
- sipProfileName: Standard SIP Profile
- useTrustedRelayPoint: Default

Default CTI Remote Device Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation
- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: {{ input.firstName }} {{ input.lastName }} CTI Remote Device
- deviceMobilityMode: Default
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- deviceTrustMode: Not Trusted
- locationName: {{ macro.CUCM_PHONE_locationName }}
- packetCaptureMode: None
- product: CTI Remote Device
- protocol: CTI Remote Device
- protocolSide: User
- useTrustedRelayPoint: Default

Default CUCM Line Template

applies to `device/cucm/Line`

Default CFT values:

- callForwardAll: {'callingSearchSpaceName': '{{ macro.CUCM_LINE_shareLineAppearanceCssName }}'}
- routePartitionName: {{ macro.CUCM_LINE_routePartitionName }}
- shareLineAppearanceCssName: {{ macro.CUCM_LINE_shareLineAppearanceCssName }}
- usage: Device

Default CUCM Phone Template

applies to `device/cucm/Phone`

Default CFT values:

- builtInBridgeStatus: Default
- callingSearchSpaceName: ((input.Phone.callingSearchSpaceName == None)) <{{macro.CUCM_PHONE_callingSearchSpaceName}}> <{{input.Phone.callingSearchSpaceName}}>
- certificateOperation: No Pending Operation

- class: Phone
- commonPhoneConfigName: Standard Common Phone Profile
- description: Created by default template
- deviceMobilityMode: On
- devicePoolName: {{ macro.CUCM_PHONE_devicePoolName }}
- lines: {'line': [{'display': '{{ input.firstName }}'}]}
- locationName: {{ macro.CUCM_PHONE_locationName }}
- packetCaptureMode: None
- phoneTemplateName: Standard 9971 SIP
- product: Cisco 9971
- protocol: SIP
- protocolSide: User
- securityProfileName: Cisco 9971 - Standard SIP Non-Secure Profile
- useTrustedRelayPoint: Default

Default CUCM RD Template

applies to `device/cucm/RemoteDestination`

Default CFT values:

- answerTooLateTimer: 19000
- answerTooSoonTimer: 1500
- delayBeforeRingingCell: 4000

Default CUCM RDP Template

applies to `device/cucm/RemoteDestinationProfile`

Default CFT values:

- callInfoPrivacyStatus: Default
- class: Remote Destination Profile
- description: Created by default template
- devicePoolName: {{macro.CUCM_PHONE_devicePoolName}}
- product: Remote Destination Profile
- protocol: Remote Destination
- protocolSide: User

Default CUCM User Template

applies to `device/cucm/User`

Default CFT values:

- enableMobility: ((fn.is_none_or_empty pwf.QAS_INPUT.existing.snr == fn.false)) <{{ fn.force_null }}> ((fn.is_none_or_empty pwf.QAS_INPUT.snr == fn.false)) <((pwf.QAS_INPUT.snr == fn.true))> ((fn.is_none_or_empty input.snr == fn.true)) <{{ fn.false }}> <((input.snr == fn.true))>
- presenceGroupName: Standard Presence group
- primaryExtension: {'pattern': '((fn.is_none_or_empty pwf.QAS_INPUT.lines.0.directory_number == fn.false)) <{{ pwf.QAS_INPUT.lines.0.directory_number }}> <{{ fn.drop }}>', 'routePartitionName': '{{ macro.CUCM_LINE_routePartitionName }}'}

Default Webex User Template

applies to `device/webex/User`

Default CFT values:

- description: AddWebEx

Reference Pexip Conference Template

applies to `relation/PexipConference`

Default CFT values:

- enable_chat: default
- name: {{ input.userid }}
- call_type: audio

10.4. Named Macro Reference

10.4.1. Named macros available to administrators

The macros that are available to administrators can be seen at two sources:

- On the tabs of the Site Defaults doc
- In configuration templates that are available to administrators for inspection, cloning, and customization

The lists group macros in the same categories as they are in the tabs on the user interface for the Site Defaults doc, while the the macros that are only in configuration templates are listed together.

10.4.2. Named macros - site defaults general

- **macro.CUCM_UDT_devicePool**

  ```
  {{ data.SiteDefaultsDoc.defaultDP }}
  ```

- **macro.CUCM_PHONE_locationName**

  ```
  {{ data.SiteDefaultsDoc.defaultLOC }}
  ```

- **macro.SITE_USP_PROFILE**

  ```
  {{ data.SiteDefaultsDoc.defaultuserprofile
     || direction:local }}
  ```

  The additional specified ensures that a user profile value is only searched for in a Site Defaults Doc at the current hierarchy, so that the hierarchy at which the Site Defaults Doc is available is limited more strictly.

- **macro.CUCM_HPILOT_routePartitionName**

  ```
  {{ data.SiteDefaultsDoc.defaulthppt }}
  ```

- **macro.CUCM_CPUG_routePartitionName**

  ```
  {{ data.SiteDefaultsDoc.defaultcpupt }}
  ```

- **macro.CUCM_CPARK_routePartitionName**

  ```
  {{ data.SiteDefaultsDoc.defaultcppt }}
  ```

- **macro.CUCM_MEETME_routePartitionName**

  ```
  {{ data.SiteDefaultsDoc.defaultmmpt }}
  ```

- **macro.DEFAULT_CUCM_GROUP**

  ```
  {{ data.SiteDefaultsDoc.defaultcucmgroup }}
  ```

- **macro.CUCM_RDP_callingSearchSpaceName**

  ```
  {{ data.SiteDefaultsDoc.default_rdp_css }}
  ```

- **macro.CUCM_RDP_devicePoolName**

  ```
  {{ data.SiteDefaultsDoc.defaultDP }}
  ```

- **macro.CUCM_RDP_dndOption**

  ```
  {{ data.SiteDefaultsDoc.default_cucm_rdp_dndoption }}
  ```

10.4.3. Named macros - site defaults device

- **macro.CUCM_PHONE_product**

  ```
  {{ data.SiteDefaultsDoc.default_cucm_phone_product }}
  ```

- **macro.CUCM_PHONE_protocol**

  ```
  {{ data.SiteDefaultsDoc.default_cucm_phone_protocol }}
  ```

- **macro.CUCM_PHONE_securityProfileName**

  ```
  {{ data.SiteDefaultsDoc.default_cucm_phone_securityprofile }}
  ```

- **macro.CUCM_PHONE_softkeyTemplateName**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_softkey }}`

- **macro.CUCM_PHONE_sipProfile**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_sipprofile }}`

- **macro.CUCM_PHONE_presenceGroupName**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_presencegroup }}`

- **macro.CUCM_PHONE_commonDeviceConfigName**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_commondeviceconfig }}`

- **macro.CUCM_PHONE_lines_line_e164Mask**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_e164_mask }}`

- **macro.CUCM_PHONE_callingSearchSpaceName**

  `{{ data.SiteDefaultsDoc.defaultdevicecss }}`

- **macro.CUCM_DP_product**

  `{{ data.SiteDefaultsDoc.default_cucm_dp_product }}`

- **macro.CUCM_DP_protocol**

  `{{ data.SiteDefaultsDoc.default_cucm_dp_protocol }}`

- **macro.CUCM_DP_phoneTemplateName**

  `{{ data.SiteDefaultsDoc.default_cucm_dp_template }}`

- **macro.CUCM_DP_lines_line_e164Mask**

  `{{ data.SiteDefaultsDoc.default_cucm_dp_e164_mask }}`

- **macro.CUCM_DP_emccCallingSearchSpace**

  `{{ data.SiteDefaultsDoc.default_dp_emcc_css }}`

- **macro.CUCM_RDP_rerouteCallingSearchSpaceName**

  `{{ data.SiteDefaultsDoc.default_rdp_rr_css }}`

- **macro.CUCM_RDP_lines_line_e164Mask**

  `{{ data.SiteDefaultsDoc.default_cucm_rdp_e164_mask }}`

- **macro.CUCM_PHONE_devicePoolName**

  `{{ data.SiteDefaultsDoc.defaultDP }}`

- **macro.CUCM_PHONE_enableExtensionMobility**

  `{{ data.SiteDefaultsDoc.default_cucm_phone_enableEM }}`

10.4.4. Named macros - site defaults line

Note: These macros will also reference a workflow variable `move_to_hn_pkid` if it is set. The variable is used in the default "Move User" workflows. If the variable value is available, it is used as the target hierarchy for macro execution.

If the macros are used in workflows outside of the default "Move User" workflows, set the variable - for example when using the macros in Line Configuration Templates.

- **CUCM_LINE_presenceGroupName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.default_cucm_line_presencegroup }
↪}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪default_cucm_line_presencegroup
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **macro.CUCM_LINE_shareLineAppearanceCssName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_vmprofile**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.default_cucm_line_vmprofile }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪default_cucm_line_vmprofile
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **macro.CUCM_LINE_routePartitionName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinept  }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinept
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardAll_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfacss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfacss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardAll_secondaryCallingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlineseccss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlineseccss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardAlternateParty_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfapcss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfapcss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardBusyInt_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfbicss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfbicss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardBusy_callingSearchSpaceName-2**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfbcss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfbcss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNoAnswerInt_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfnaicss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfnaicss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNoAnswer_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfnacss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfnacss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNoCoverageInt_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfncicss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfncicss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNoCoverage_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfnccss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfnccss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNotRegisteredInt_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfnricss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfnricss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardNotRegistered_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfnrcss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfnrcss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

- **CUCM_LINE_callForwardOnFailure_callingSearchSpaceName**

```
(( fn.is_site == fn.true ))<{{ data.SiteDefaultsDoc.defaultlinecfofcss }}>
(( fn.is_none_or_empty pwf.move_to_hn_pkid == fn.false ))<{{ data.SiteDefaultsDoc.
↪defaultlinecfofcss
        | __hierarchy_friendly_path: pwf.move_to_hn_pkid }}><>
```

10.4.5. Named macros - site defaults user

- **macro.DEFAULT_USER_ROLE**

```
(( data.SiteDefaultsDoc.defaultuserrole == None ))
   <selfservice> <{{ data.SiteDefaultsDoc.defaultuserrole}}>
```

If a default user role is not specified in the Site Defaults Doc, then the role value is the *selfservice* role.

- **macro.CUCM_USER_presenceGroupName**

```
{{ data.SiteDefaultsDoc.default_cucm_user_presencegroup }}
```

- **macro.CUCM_USER_serviceProfile**

```
{{ data.SiteDefaultsDoc.default_cucm_user_serviceprofile }}
```

- **macro.CUCM_USER_subscribeCallingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecss }}
```

10.4.6. Named macros - site defaults CUC

- **macro.CUC_PHONE_SYSTEM**

```
{{ data.SiteDefaultsDoc.defaultcucphonesystem }}
```

- **macro.CUC_USER_templateAlias**

```
{{ data.SiteDefaultsDoc.defaultcucsubscribertemplate }}
```

- **macro.CUC_SMPP_PROVIDER**

```
{{ data.SiteDefaultsDoc.defaultcucsmppprovider }}
```

10.4.7. Named macros - site defaults Hotdial

- **macro.CUCM_DP_defaultplarcss**

  `{{ data.SiteDefaultsDoc.defaultplarcss }}`

- **macro.CUCM_HOTDIAL_defaultplarcss**

  `{{ data.SiteDefaultsDoc.defaultplarcss }}`

10.4.8. Named macros in configuration templates

- **macro.CUC_HTML_NotificationTemplateName**

```
{{ data.SiteDefaultsDoc.defaultcuchtmlnotificationtemplate
   | name:macro.DPSITE }}
```

  By default, the macro: `DPSITE` resolves to the site name at the current hierarchy, in other words: `{{ data.BaseSiteDAT.SiteName }}`

  The macro therefore resolves to the CUC notification template with the same name as the current site.

- **macro.CUC_DEFAULT_USER_TEMPLATE**

  `Standard Common Phone Profile`

- **macro.CUCM_LINE_callForwardBusy_callingSearchSpaceName**

  `{{ fn.evaluate macro.CUCM_LINE_callForwardBusy_callingSearchSpaceName-2 }}`

  The macro uses a macro function call to resolve another macro. Refer to the notes at the relevant macro.

- **macro.NEXT_AVAILABLE_LINE**

```
{{ data.InternalNumberInventory.internal_number |
    used:fn.false,available:fn.true |
    direction:up,limit:1 }}
```

  Returns the next available line from the internal number inventory.

- **macro.MappedE164fromDNLookup**

  `{{ fn.get_e164_number cft.LineX.dirn.pattern }}`

  Look up the mapped E164 number given a DN.

- **macro.E164MaskMappedorPubNum**

```
 (( fn.is_none_or_empty macro.MappedE164fromDNLookup == false ))
 <{{fn.replace macro.MappedE164fromDNLookup,\}}>
 <{{data.DpSite.pubNumber || direction:local }}>

Takes the results of ``macro.MappedE164fromDNLookup`` and checks to see
if it returned a value. If not, return the published number

Used in the various CFTs in Quick Add User - for ``device/cucm/Phone``,
``device/cucm/DeviceProfile``, ``device/cucm/RemoteDestinationProfile``.
```

- **macro.CUCM_UDT_commonPhoneProfile**

  `Standard Common Phone Profile`

- **macro.CUCM_UDT_ownerUserId**

  `Current Device Owner's User ID`

- **macro.CUCM_UDT_phoneButtonTemplate**

  `Universal Device Template Button Layout`

- **macro.SITENAME**

  `{{ data.VOSS-Site-DialPlan.SiteName }}`

  During site creation, an instance of `data/VOSS-Site-DialPlan` is created at the site hierarchy. This data model stores a reference to the name of the site.

10.4.9. Pull sync delete threshold

During a data sync where you delete an Automate resource from the device so that the key is in the Automate list but not in the 'device' list, a pull sync will remove the resource in Automate.

In order to manage the number of resources that are deleted on Automate during this process, a named macro should be added to the system, with the following values:

- `name`: *PULL_SYNC_DELETE_THRESHOLD_<device_type>*
- `macro`: number
- `hierarchy`: the hierarchy where the sync takes place.

Upon a data sync of the device type, a check for the named macro at the hierarchy will take place and local deletes will be restricted to the number defined in the macro.

Note:

- The `macro` value is a JSON string containing the number, e.g. `"macro": "20"`. This number defines the maximum number of deletes that are allowed to take place during a sync, so that if there are for example more than 20 deletes to process, then the data sync/import will fail.

  If the macro is created on the GUI, simply type in the number as value.

- The `<device_type>` value is obtained from the model name component of the `connection_parameters_type` field of the `data/DeviceModelMapping` model. Refer to the table below.

In JSON format, the named macro is for example as in the snippets below:

```
[...]
 "meta": {
   "model_type": "data/Macro",
   "hierarchy": "sys.hcs.CS-P.CS-NB.AAAGlobal",
   "tags": []
 },

 "data": {
```

(continues on next page)

(continued from previous page)

```
   "macro": "20",
   "name": "PULL_SYNC_DELETE_THRESHOLD_MSTeamsOnline",
   "description": "PULL_SYNC_DELETE_THRESHOLD_MSTeamsOnline"
 }
[...]
```

The list of provided PULL_SYNC macros and their maximum number of deletes are listed below. These macros can be cloned to a lower hierarchy and adjusted if required.

Name	macro
PULL_SYNC_DELETE_THRESHOLD_CallManager	50
PULL_SYNC_DELETE_THRESHOLD_Ldap	50
PULL_SYNC_DELETE_THRESHOLD_MSExchangeOnline	50
PULL_SYNC_DELETE_THRESHOLD_MSGraph	20
PULL_SYNC_DELETE_THRESHOLD_MSTeamsOnline	20
PULL_SYNC_DELETE_THRESHOLD_Spark	50
PULL_SYNC_DELETE_THRESHOLD_UnityConnection	50
PULL_SYNC_DELETE_THRESHOLD_Zoom	50

Note: The model name following the `data/` model type is inserted into the macro name.

The table below maps the device type names as seen on the Data Sync GUI to the related model in full (remove `data/` in the macro) :

Table 1: Pull Sync Table

Device Type	Full data model name
Active Directory	`data/activedirectory`
Active Directory Hybrid	`data/activedirectoryhybrid`
Assurance Arbitrator	`data/AssuranceArbitrator`
AudioCodes	`data/AudioCodes`
AudiocodesSBC	`data/AudiocodesSBC`
AzureADOnline	`data/AzureADOnline`
BroadsoftOCIP	`data/BroadsoftOCIP`
Cisco Unified CM	`data/CallManager`
Cisco TelePresence Exchange	`data/CiscoTelePresenceExchange`
Cisco TelePresence Management Suite	`data/CiscoTMS`
Call Manager Control Center Services	`data/CmCcs`
Exchange	`data/exchange`
Exchange Hybrid	`data/exchangehybrid`
Exchange Online	`data/exchangeonline`

continues on next page

Table 1 – continued from previous page

Device Type	Full data model name
Call Manager AXL Generic Driver	data/GenericCucm
Unity Connection GUI	data/GuiCuc
CallManager GUI	data/GuiCucm
IMP GUI	data/GuiImp
Cisco HCM-F	data/Hcmf
HTTP	data/Http
IMAP4	data/Imap
IOS	data/Ios
LDAP	data/Ldap
MSExchangeOnline	data/MSExchangeOnline
MSGraph	data/MSGraph
Microsoft Online	data/msonline
MSOperatorConnect	data/MSOperatorConnect
MSTeamsOnline	data/MSTeamsOnline
Netflow	data/Netflow
Notification Service	data/NotificationService
OracleECB	data/OracleECB
OracleSBC	data/OracleSBC
Pexip Infinity Conferencing Platform	data/Pexip
PGW	data/Pgw
PowerShell	data/powershell
PRSCallControl	data/PRSCallControl
Roomos	data/Roomos
ServiceNowOnline	data/ServiceNowOnline
sfb2015	data/Sfb2015
SkypeForBusiness	data/skypeforbusiness
SkypeForBusiness Hybrid	data/skypeforbusinesshybrid
SkypeForBusiness Online	data/skypeforbusinessonline
SMTP	data/Smtp
Service Now	data/snow
Spark	data/Spark
Cisco Unified Contact Center Express	data/Uccx
Cisco Unity Connection	data/UnityConnection
Cisco WebEx	data/WebEx
Zoom	data/Zoom

If during a sync the macro is available and the number value in the macro is reached while there are more instances, the sync log will also show an error. If no macro is defined as above, then the pull sync will remove all local instances on Automate.

11. Macro Reference

11.1. Macros

Tip: *Use the Action search to navigate Automate*

Macros are used to return data from the system in various formats, to test for conditions, map data from Admin Portal or bulk loader input to various elements in the system (in conjunction with configuration ltemplates) and to access data in workflow steps.

Note: You can view existing macros and add new macros via the **Macros** page. For details on creating macros, see the Advanced Configuration Guide.

Various macro functions are available. These serve as boolean operators or can be used to carry out various numeric functions, string manipulation functions, list functions, time functions, and hierarchy related functions.

Note: The following macro functions are only available to administrators with an Access Profile set to **Full Access**:

- `fn.request_get`
- `fn.perform_http_get`

Macros can be created for re-use, named and stored as an instance of the Macro data model. When re-used, the reference prefix syntax is of the format: *macro.<macroname>*.

Named macros and macro functions can be nested within other macros.

Refer to the Macro Reference topics.

11.2. Macro Syntax

11.2.1. Macro syntax brackets

Macros can have any of the following markup:

- `{{` and `}}`

indicate macros that resolve to single values. The value can also return an object. A direction parameter is also available for hierarchy searching. This is indicated by `||`. Refer to the topic on SELECT-FROM-WHERE type macros for details.

Any number of single opening and closing brackets (`{` and `}`) can also occur inside these scalar macros.

- `{#` and `#}`

  indicate macros that resolve to lists of values.

Note: Lists of values can be obtained form `relation.`-type models include the left model values as well as `pkid` values, so that relation models can be used in macro queries.

- `{%` and `%}`

  indicate macros that resolve to dictionaries

- `((` and `))`

  indicate macros that test for a condition are enclosed in round brackets ((and)) - these macros evaluate to True or False

  The comparison operators that are available for these macros are: `==`, `!=`, `<`, `>`, `<=`, `>=`.

  The 'SELECT FROM WHERE' operator can be used in a test is `|`, for example:

```
(( data.User.username | username:John == 'John' )) <true> <false>
```

  See also 'SELECT FROM WHERE'-type macros below.

  It is *highly* recommended to use named macros for things in this case, in other words, instead of:

```
(( data.User.username | username:John == 'John' )) <true> <false>
```

  use:

```
MY_USER = {{ data.User.username | username:input.username }}
(( macro.MY_USER == input.username )) <true> <false>
```

  The WHERE clause can also contain a logical AND, represented by a comma (`,`), for example:

```
(( device.cucm.Line.pattern | pattern:pwf.PassedLine.pattern, routePartitionName:pwf.
↪PassedLine.routePartitionName == '' ))
```

  Similarly, the test condition can contain a logical AND, for example:

```
(( fn.list_contain macro.GS_CLIMaskANANumberType_MCR,macro.GS_
↪CLIMaskANAGetAllowedTypes_MCR == true ))
```

- `((test)) <if value> <else value>`

  IF ELSE type conditional macro.

- `((test)) <value> ((test)) <value> <value>`

  IF ELSEIF ELSE-type macros combine tests and result values if the test resolves to True or False. The logic is IF (test) THEN <value> ELSE IF (test) THEN <value> ELSE <value>.

  Example:

```
((self.a == self.b)) <foo-{{CallManager.host}}>
 ((self.b ==  self.c)) <foo-{{CallManager.username}}>
 <foo-{{CallManager.version}}>
```

This macro tests for the equality of values in a calling model (referenced by 'self') and returns an evaluation for the condition that is True. The evaluation refers in dot notation to attributes of a Data Model called 'CallManager' and concatenates the result with a string 'foo-'.

- 'SELECT FROM WHERE'-type macros returning single-, dictionary- and list type values and can take parameters.

  The format is:

  - *{{ SELECT FROM | WHERE }}*
  - *{% SELECT FROM | WHERE %}*
  - *{# SELECT FROM | WHERE #}*

  These types and their parameters are illustrated in the topic on 'SELECT FROM WHERE' Macros: *SELECT FROM WHERE macro syntax*.

11.2.2. Macro syntax multiline

Macros can contain line breaks. The Admin Portal provides a multiline input box for macros. For example:

- If a macro that returns a string has string prefix or suffix, for example, a macro with the value:

```
"AAA
{{data.Countries.iso_country_code | country_name:'South Africa'}}
AAA"
```

will output:

```
AAA
ZAF
AAA
```

A JSON export of the macro will show the line breaks, for example:

```
"data": {
  "macro": "AAA\n{{data.Countries.iso_country_code | country_name:'South Africa'}}\nAAA",
  "name": "MACRO1"
}
```

- If the macro is an Expect script:

```
expect "HQ>\r"
send "enable\r"
expect "Password:\r"
send  "{{device_details.enable_password}}\r"
```

Line breaks cannot be entered inside a macro, for example:

```
"AAA
{{data.Countries.iso_country_code |
country_name:'South Africa'}}
AAA"
```

is not valid.

11.2.3. Dot notation

Overview

Dot notation is used to reference macro functions, model attributes, defined variables, step references, and non-attribute values in the model schema.

Dot notation is a structured way to access:

- Macro functions
- Model attributes
- Defined variables
- Workflow steps
- Non-attribute metadata

It is a pathing syntax that lets you drill into specific values or functions within the platform's data models and templates.

Syntax patterns and usage

- `fn.macrofunction_name`

  Calls a macro function. See the **Macro Functions** section in the Advanced Configuration Guide.

- `self.attribute`

  Refers to the current model's attribute.

  Note: `attribute` must be a valid attribute name of the calling model (typically within a configuration template).

- `previous.attribute`

  Retrieves the previously saved value of an attribute. Useful when comparing against the current value during model updates.

- `input.attribute`

  Identifies values provided via GUI input, bulk load, provisioning workflow foreach variable or context.

- `pwf.variablename`

  References a variable defined in a provisioning workflow step.

- `cft.attribute`

  Access values from the current `context_var` of a `foreach` loop within a configuration template.

- `modelname.attribute`

  Defaults to referencing the data model type.

- `modeltype.modelname.attribute`

  Used for non-data model types.

- `modeltype.modelname.attribute.NUMBER`

  Refers to a specific indexed attribute (zero-based), referring to attribute NUMBER-1 where there is more than one attribute.

  Example: `modeltype.modelname.attribute.0` accesses the first item. See section *Macro examples*. You can also use a wildcard:

```
{# device.cucm.Phone.lines.line.\*.dirn #}
```

- `macro.name`

  References a defined macro by name.

- `workflow.stepSTEPNUMBER.pkid`

  Overrides the context hierarchy by specifying the `pkid` of a workflow step.

  Example: `workflow.step2.pkid`

Non-attribute notation

These metadata (special) fields are used for hierarchy and identification and can be accessed using dot-notation:

- `__pkid`
- `__bkey`
- `__hierarchy`
- `__hierarchy_friendly_path`
- `__hierarchy_friendly_parent_path`

Examples:

```
{# data.Countries.__pkid #}
{# data.CallManager.__bkey #}
{{ data.CallManager.__bkey | host:172.29.248.150 }}
{{ data.CallManager.__hierarchy | host:172.29.248.150 }}
{{ data.CallManager.__hierarchy_friendly_path | host:172.29.248.150 }}
{{ data.CallManager.__hierarchy_friendly_parent_path | host:172.29.248.150 }}
```

Including the unique ID for each instance of special fields

You can use a wildcard (*) appended to `__pkid` with any of the special (metadata) fields in a macro query to return all fields plus the unique identifier for each instance.

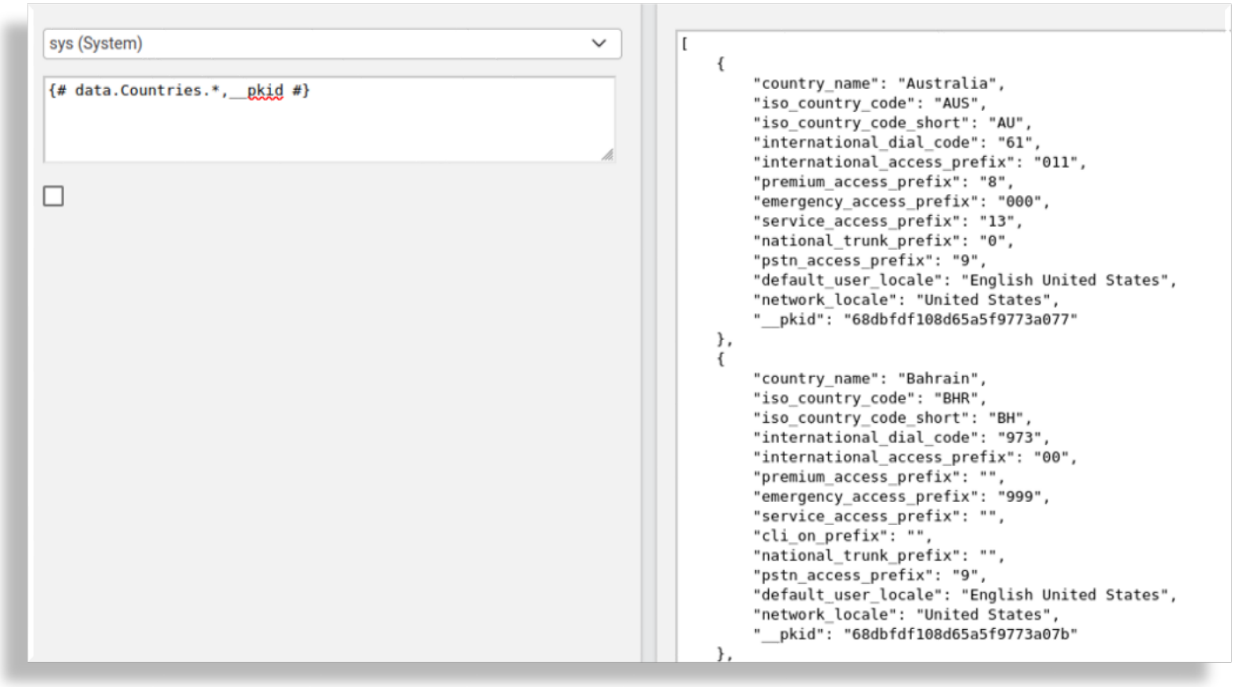
Examples:

- `{# data.Countries.*,__pkid #}`
- `{# data.Countries.*,__pkid,__hierarchy_friendly_path,__hierarchy,__bkey #}`

A unique ID is required for each instance when using a wildcard since it is not possible to rely on all fields to be unique. This is useful for overbuild workflows or any scenario requiring unique identification.

For example, macro `{# data.Countries.*,__pkid #}` returns all countries in the database, plus the pkid. This syntax ensures that the macro output includes all fields and the unique identifier for each instance.


```
{
     "country_name": "Australia",
     "iso_country_code": "AUS",
     "iso_country_code_short": "AU",
     "international_dial_code": "61",
     "international_access_prefix": "011",
     "premium_access_prefix": "8",
     "emergency_access_prefix": "000",
     "service_access_prefix": "13",
     "national_trunk_prefix": "0",
     "pstn_access_prefix": "9",
     "default_user_locale": "English United States",
     "network_locale": "United States",
     "__pkid": "63f7885cc33191b9204fa73e"
 },
```

Combining hierarchy fields

The `hierarchy` fields can be combined with model attribute names in a comma separated list, for example:

```
{# data.Countries.country_name,__hierarchy,__hierarchy_friendly_path #}
```

Output snippet:

```
[
    {
        "country_name": "Australia",
        "__hierarchy": "58d303503f44d58341d61775",
        "__hierarchy_friendly_path": "hcs"
    },
    {
        "country_name": "Bahrain",
        "__hierarchy": "58d303503f44d58341d61775",
        "__hierarchy_friendly_path": "hcs"
    },

 ...
```

Note: Meta attribute properties can also be used in a macro filter. See *Macro Syntax to Filter by Meta Properties*.

Sequence and looping

- To indicate sequence instance with `SEQ` - the value is a loop sequence number starting from 1 or a wizard step number:
  - It is obtained from a **Foreach List Macro** in a provisioning workflow or a **Foreach Elements** loop in a configuration template.

    This value can be used to refer to an *attribute* of a model. An array item in a configuration template has a **Foreach Elements** loop with a variable `phoneX`:

    ```
    {# self.Phone.{{fn.subtract input.phoneX.SEQ,1}}.lines.line #}
    ```

    that refers to an attribute (line in a list of phones), starting with the first one:

    `self.Phone.0.lines.line`
  - This value can be used to refer to a *Wizard step* (stepSTEPNUMBER) in its configuration template. A **Foreach Elements** loop with a variable `step` that holds a list of STEPNUMBER obtained from the wizard:

    ```
    {# input.define_wizard_steps #}
    ```

    so that stepSTEPNUMBER can be incremented with:

    `step{{cft.step.SEQ}}`

11.2.4. SELECT FROM WHERE macro syntax

The types of return values are indicated by the syntax:

- *{{SELECT FROM | WHERE}}* for single values or objects
- *{%SELECT FROM | WHERE%}* for dictionaries
- *{#SELECT FROM | WHERE#}* for lists

The `SELECT FROM` part is a model reference and uses dot notation.

For lists and dictionaries, `SELECT FROM` notation can contain a wild card asterisk * to return all matching values.

Note: See:

- *Non-attribute notation* section in *Dot notation* in the Advanced Configuration Guide

---

Examples returning all User attributes:

```
{# data.User.* |username: js54321, last_name: 'van Dever' #}
{% data.User.* |username: js54321 %}
```

The WHERE part is one or more comma-separated name:value pairs of attribute values of the model reference. If the value itself has a comma, it should be wrapped in quotes.

For example:

```
{{ data.Address.*  | street_name: 'The Willows, Park Crescent' }}
```

The value can also contain:

- a reference to an *empty* field, as in:

  ```
  {# data.User.* | email: null #}
  ```

- a reference to a defined macro used to return a value, as in:

  ```
  {# data.DRTPBXMeta.* | PBX:macro.getHost #}
  ```

- a boolean value using the corresponding macro function, as in:

  ```
  {{ data.DATA1.name | code: fn.true }}
  ```

The WHERE part also supports:

- Asterisk * for filtering.

  Examples:

  ```
  {{ device.cucm.Phone.lines.line | name: "SEPD13B004F0719",
     lines.line.*.dirn.pattern:"1006",
     lines.line.*.dirn.routePartitionName:
      "AllowEmerCalls-NewSite4" }}
  ```

If there is a * in the WHERE clause:

- the results list is reduced with that specific clause
- the specific WHERE clause is a list

Note that only one * is supported and if the WHERE clause has an invalid field name, it will be ignored and still return the data.

- regular expression type syntax:

  `field:/regex/`

  Note: regular expression syntax should not be used when querying collections with more than 500 members.

  Examples:

```
{# data.Countries.country_name | country_name:/ia$/ #}
```

  returns names that end in “ia”:

```
[ "Australia", "Saudi Arabia" ]
```

  To exclude a list of countries matching “ia” in the name:

```
{# data.Countries.country_name | country_name:/[^ia]$/ #}
```

  To carry out a case-insensitive search, add the regex parameter:

```
{# data.UserSavedSearch | username: /input.username/i #}
```

  Will for example match:

```
CS-PADMIN@csp.com
CS-PAdmin@csp.com
cs-padmin@csp.com
```

  Macros cannot be nested in the regex. This will *not* work:

```
{# data.Countries.* | iso_country_code: /[^({{input.ISO}})]/ #}
```

  But this will work:

  Macro ISO_REGEX:

```
/[^({{input.ISO}})]/
```

  Macro:

```
{# data.Countries.* | iso_country_code:macro.ISO_REGEX #}
```

- Attributes not in the data of the model schema
  - `__pkid`
  - `__device_pkid`

- `__bkey`
- `__hierarchy`

  Example format:

  - 24-character hierarchy pkid
  - dotted path name
- `__hierarchy_friendly_path`
- `__hierarchy_friendly_parent_path`

Note: See:

- *Non-attribute notation* section in *Dot notation* in the Advanced Configuration Guide

---

Examples:

- Given the following macro `USA_pkid`:

```
{{ data.Countries.__pkid | iso_country_code:USA }}
```

  then macro:

```
{{ data.Countries.country_name | __pkid:macro.USA_pkid }}
```

  will return:

```
{"United States of America"}
```

- Using `__device_pkid`:

```
{{ device.cucm.CallManager.__device_pkid | name:CM_sol-cucm-ob-01}}
```

  returns:

```
5c6f4ad0de894e0014e5b84c
```

Note that when using `__hierarchy_friendly_path` in a WHERE clause, the option clause *direction* will be ignored, for example:

```
{# data.Countries.* | __hierarchy_friendly_path: sys.TestMacros #}
```

will only return Countries at this hierarchy node.

For more details on this parameter, refer to the topic Macro Syntax to Filter by Meta properties.

The SELECT-FROM-WHERE macros can also take additional filter parameters that restrict results:

- `| direction:[up|down|local|parent|below|above|fulltree]`
- `| device:[pkid of device]`
- `| ndl:[pkid of ndl]`
- `| limit:[number]`

- `| skip:[number]`
- `| title:[character]`

The *direction* option can be added to return values relative to the current hierarchy position. The default direction is *down*:

- `direction:up` - Upwards. Include current hierarchy.
- `direction:down` - Downwards. Include current hierarchy.
- `direction:local`: - On this level only. Include current hierarchy.
- `direction:parent` - Parent only. Exclude current hierarchy, in other words, search the parent as `local`.
- `direction:below` - Downwards. Exclude current hierarchy.
- `direction:above` - Upwards. Exclude current hierarchy if current hierarchy is not top (`sys`).

  This is the *default* hierarchy search result restriction if no *direction* option is specified. In this case, *only the first value above* will be returned.

  Note: It is best practice to *always* use the *direction* option.

- `direction:fulltree` - Upwards (`up`) *and* Downwards (`down`) from the current hierarchy. Allows for a full hierarchy search.

It is recommended to use a *direction* option in a list macro without a WHERE clause, for example:

```
{# data.Countries.* || direction:up #}
```

In a 'SELECT-FROM-WHERE'-type macro, a single bar indicates the direction, for example:

```
{{ data.Countries | iso_country_code:AUS | direction:up }}
```

If used in other macro types, a double bar is used for parameters, for example:

```
{# data.SiteDefaultsDoc.defaultcucphonesystem || direction:local #}
```

When traversal is `up` or `above`, results will be ordered starting with ones at the lowest hierarchies. Otherwise, results will be ordered starting with the ones at the highest hierarchies. Results at the same hierarchy will be in arbitrary order. Also refer to the optional *to* specifier below.

Added to the *direction* option is:

- an optional *limit* specifier. When used, the results returned by a list comprehension will be limited to the specified count, for example:

  ```
  {# data.test_user.name || direction:above,limit:2 #}
  ```

  This will return the first two names of `data/test_user` instances at the closest ancestors.

  By default, for the following filter specifiers values apply to lists if they are not present:

  - `skip` (default: 0) - skip over a number of values before listing
  - `limit` (default: 2000) - number of values to return in the list

  So, if the first list macro was:

```
{# data.test_user.name || skip:0,limit:2000 #}
```

then the second batch of results can be obtained by:

```
{# data.test_user.name || skip:2000,limit:2000 #}
```

- an optional *to* specifier to restrict the *direction:up* or *direction:down* specifier to a specified hierarchy, for example:

```
{# data.Countries.* || direction:up,to:Customer #}
```

This example will restrict the search upwards up to the Customer hierarchy.

```
{# data.Countries.* || direction:down,to:Site #}
```

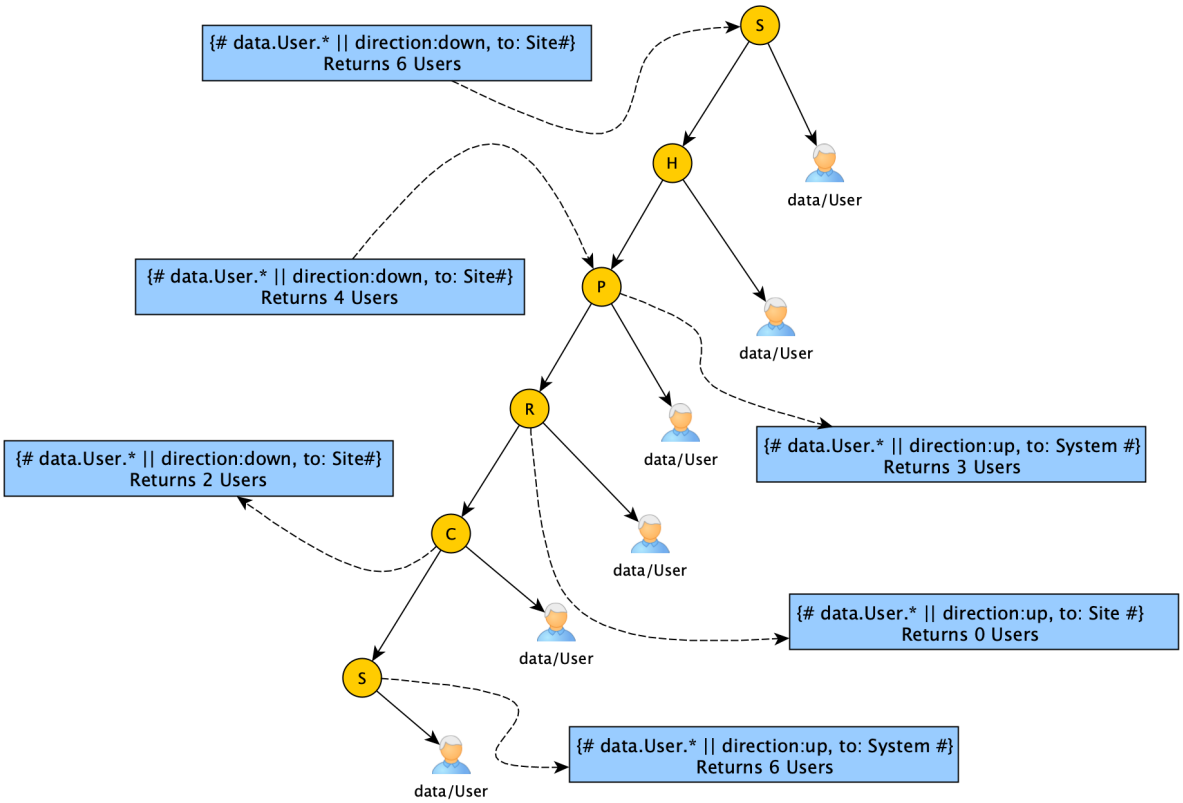
This example will restrict the search downwards up to the Site hierarchy.

Valid hierarchy values for `to:` are:

  - `System`
  - `Hcs`
  - `Provider`
  - `Reseller`
  - `Customer`
  - `IntermediateNode`
  - `Site`
  - `LinkedSite`

All data is returned in the case of incorrect spelling or invalid hierarchy levels.

Example showing query results:



If a list macro is used to provide a list of input values on the GUI, note that custom values are permanently allowed. This means that a custom value can be entered in the GUI input. Adding a GUI rule on the GUI form that sets an object property attribute “Allow Custom Values” to false, will not affect this functionality.

In addition, a `title` filter can be applied if the *SELECT-FROM* query is for a string to only return values matching its value, with a regular expression, for example:

```
{# data.Countries.country_name || title:.*a$ #}
```

returns:

```
[
 "Australia",
 "Canada",
 "China",
 "Saudi Arabia",
 "South Africa",
 "United States of America"
]
```

Device option:

A device or ndl (Network Device List) pkid can be specified to restrict a query, for example, assume a named macro MY_CUCM_PKID_150:

```
{{ data.CallManager.__pkid | host:172.29.248.150,port:8443 }}
```

then we can select from the specified device as follows:

```
{{ device.cucm.HuntList.__pkid | name: "DR-Test1" | device:macro.MY_CUCM_PKID_150 }}

{{ device.cucm.HuntList.__hierarchy | name: "DR-Test1" | device:macro.MY_CUCM_PKID_150 }}

{{ device.cucm.HuntList.__hierarchy_friendly_path | name: "DR-Test1" |
  device:macro.MY_CUCM_PKID_150 }}

{{ device.cucm.HuntList.__hierarchy_friendly_parent_path | name: "DR-Test1" |
  device:macro.MY_CUCM_PKID_150 }}
```

In a GUI Rule, [and] indicate references to values in the current usage context of the GUI Rule if the macro is added as a **Value** to the GUI Rule.

A GUI rule **Action** can also have an API call as its **Source**. The references are current context hierarchy pkid's or to field attribute names in the WHERE section of SELECT FROM WHERE-type macros - enclosed in square brackets [ ].

For example:

```
/api/tool/Macro/?method=evaluate&hierarchy=[hierarchy]&input={{Countries.iso_country_
↪code |
   country_name:[countries.name]}}
```

The syntax in a GUI Rule for a Wizard also uses [], in the format *[stepData.STEPNAME.ATTRIBUTE]*, for example:

```
[stepData.SubscriberType.role]
```

11.2.5. Macro nesting

To nest macro calls, create a named macro. Nesting inside macros is not supported.

For example, the following incorrect macro:

```
(( fn.list_in Kit, {# fn.split {{ fn.one device.ldap.user.streetAddress |
    sAMAccountName:bjones }} #} == True))
```

should be split up into named macros:

- macro: LDAP_USER

  ```
  {{ fn.one device.ldap.user.streetAddress | sAMAccountName:bjones }}
  ```

- macro: LDAP_USER_ADDRESS_LIST

  ```
  {# fn.split macro.LDAP_USER #}
  ```

So the correct macro usage should be:

```
(( fn.list_in Kit,macro.LDAP_USER_ADDRESS_LIST == True))
```

11.2.6. Macro syntax to filter by meta properties

Macro results can also be filtered by the `meta` data of a resource.

A typical resource instance has associated `meta` data, for example:

```
meta: {
  title: "Australia - AUS"
  cached: true
  tags: [0]
  schema_version: "0.1.5"
  summary: "true"
  references: {...}-
  actions: [...12]-
  model_type: "data/Countries"
  path: [2]
  summary_attrs: [...3]-
  business_key: {...}-
  tagged_versions: [0]
}
```

The following macro fields are supported to filter by these properties:

- `__meta.business_key`
- `__meta.model_type`
- `__meta.schema_version`
- `__meta.system_resource`
- `__meta.tags`
- `__meta.title_format`
- `__meta.uri`
- `__meta.version_tag`

The macro fields can be combined with model attribute names in a comma separated, for example:

```
{# data.Countries.country_name,__meta.schema_version | country_name:Australia #}
```

Output:

```
[
    {
        "country_name": "Australia",
        "__meta": {
            "schema_version": "0.1.5"
        }
    }
]
```

As a further example: if the `system_resource` is set in the `meta` section of the resource, then the following macro can be used:

```
{# data.ConfigurationTemplate.name | __meta.system_resource: true #}
```

The output is all the Configuration Template names where it is a system resource:

```
[
  "AddFeature_Attributes_View",
  "AddFeature_DM",
  "AddFeature_DOMM",
  "AddFeature_DOMM_DM",
  "AddFeature_PKG",
  "AddFeature_PWF_Add",
  "AddFeature_PWF_Add_DM",
  "AddFeature_PWF_Del",
  "AddFeature_PWF_Del_DM",
  "AddFeature_PWF_Mod",
  "AddFeature_PWF_Mod_DM",
  "AddFeature_SelectModels_View",
  "AddWizard_GuiRules"
]
```

For devices, the following are examples of macros that are available for the device NDL:

```
__device_meta.ndl.name
__device_meta.ndl.data/CallManager.pkid
```

11.3. Macro Functions

11.3.1. Macro function syntax

Macro functions have the format: `fn.<function_name>` For an alphabetical list of macro functions, refer to the Index.

If a macro function take more than one argument, these are comma-separated.

Important: While many of the functions can take parameters separated by commas and also white space, it is recommended to *use only a comma* to separate parameters.

For example, use the parameter separator as follows:

```
{{ fn.containsStartsWith aaa,aaaffccffdd }}
```

and *not with a space*, as in:

```
{{ fn.containsStartsWith aaa, aaaffccffdd }}
```

A null parameter is indicated by no value following a comma or between commas, for example:

```
{{ fn.cucm_get_line_details 4025,,is_line_shared }}
```

11.3.2. Numeric functions

- *fn.is_int*: Return true or false if the parameter is an integer or not.
- *fn.zeropad*: Left pad a given number with zeros up to a given pad number.

Example	Output
`{{fn.zeropad 123,6}}`	`000123`

- *fn.minval*: For integers, return the minimum value of a provided list.

Example	Output
`{{fn.minval 2,3,130,1,30}}`	`1`

- *fn.maxval*: For integers, return the maximum value of a provided list.

Example	Output
`{{fn.maxval 2,3,130,1,30}}`	`130`

- *fn.add*: Add two integers.

Example	Output
`{{fn.add 2,3}}`	`5`

- *fn.subtract*: Subtract two integers.

Example	Output
`{{fn.subtract 2,3}}`	`-1`

- *fn.multiply*: Multiply two integers.

Example	Output
`{{fn.multiply 2,3}}`	`6`

- *fn.divide*: Divide two integers.

Example	Output
`{{fn.divide 20,10}}`	`2`

- *fn.random*: Select a random number between two provided integers.

Example	Output
`{{fn.random 5,10}}`	`7`

11.3.3. String functions

- *fn.is_string* : Return true or false if the parameter is a string or not.
- *fn.index* : Return the i'th item of an iterable, such as a list or string.

Example	Output
`{{ fn.index 'foo bar baz',5 }}`	`'b'`

- *fn.mask* : Return a mask of (length + modifier) instances of char.

Example	Output
`{{ fn.mask X,2,3 }}`	`XXXXXX`

- *fn.length* : Return the length of a string.

Example	Output
`{{ fn.length This is a valid string }}`	`22`

- *fn.split* : Split a string by delimiter, returning a list.

Example	Output
`{# fn.split foo:bar:baz,: #}`	`['foo', 'bar', 'baz']`

- *fn.join* : Join a string by delimiter. If no delimiter is provided, the list is returned as a single string.

Example	Output
`{{ fn.join 1234,: }}` `{{ fn.join 1234 }}`	`1:2:3:4` `1234`

- *fn.title* : Return a string in title case.

Example	Output
`{{ fn.title 'foo bar baz' }}`	`'Foo Bar Baz'`

- *fn.upper* : Return an uppercase version of the input string.

Example	Output
`{{ fn.upper somevalue }}`	`SOMEVALUE`

- *fn.lower* : Return a lower case version of the input string.

Example	Output
`{{ fn.lower SOMEVALUE }}`	`somevalue`

- *fn.contains* : Return true or false if string is contained in another.

Example	Output

`{{ fn.contains needle,haystack }}`
`{{ fn.contains hay,haystack }}`
`(( fn.contains Kit,1234 Kit Creek == True ))`
	`false`
	`true`
	`true`

- *fn.sub_string* : Return the substring of a string from a start to an end position.

Example	Output
`{{ fn.sub_string haystack,0,2 }}` `{{ fn.sub_string haystack,7,7 }}`	`hay` `k`

- *fn.containsIgnoreCase* : Return true or false if string is contained in upper- or lower case.

Example	Output
`{{ fn.containsIgnoreCase aaa,bbbaAaccc }}`	`true`

Example	Output

Input
`{"input": {"UserName": "user@host.org"}}`
Function call
`((fn.containsIgnoreCase Host,input.UserName == True))`
`<{{input.UserName}}><>`
	`user@host.org`

- *fn.containsStartsWith* : Return true or false if source string is the start of target string.

Example	Output
{{ fn.containsStartsWith aaa,aaaffccffdd }}	true

- *fn.containsStartOf* : Return true or false if start of source string is target string.

Example	Output
{{ fn.containsStartOf ffnnccgg,ffnn }}	true

- *fn.regex_match* : Return true or false if the regular expression matches the start of target string.

Example	Output
{{ fn.regex_match a,ab }}	true
{{ fn.regex_match a$,ab }}	false
{{ fn.regex_match \d+,abc555 }}	false
{{ fn.regex_match \d+,555abc }}	true

- *fn.isexactly* : Return true or false if source string is exactly the same as target string.

Example	Output
{{ fn.isexactly source1,source1 }}	true

- *fn.replace* : Replace target substring for source substring in source string.

Example	Output
Note: no spaces between commas. `{{ fn.replace dddddAAAc,AAA,FFF }}`	`dddddFFFc`

- *fn.validate_name* : Return true or false if the string as well as its stripped content (default white space removed) exists, i.e. is then more than 0 characters.

Example	Output

`{{ fn.validate_name }}`
 	
`{{ fn.validate_name a }}`
	`false`
	
	`true`

- *fn.fix_non_ascii* : Given a string containing non-ASCII characters, this function applies 3 steps of processing and returns a string with all non-ASCII characters mapped to ASCII characters.

  The first step can be controlled by the optional “from” and “to” characters parameter pair that return a string with “from” characters replaced with “to” characters. This allows for control over the replacement.

  The 3 steps are carried out in the following sequence:

  1. Map characters in the given string by using the corresponding characters in the “*<from>,<to>*” string parameter pair.
     - If the parameters are not added, this step is skipped.
     - If it is required to have control over the mapping of non “accented” characters (see step 2), then this default mapping can be done by providing the “*<from>,<to>*” string parameter pair.
  2. Map all “accented” (accent, circumflex, diaeresis, etc.) non-ASCII characters to ASCII characters.
  3. Map all remaining non-ASCII characters with the underscore character: _.

  For example:

  - string = `Ŧest Nón Asciï`
  - *<from>,<to>* = `ï,G`
  - result = `_est Non AsciG`

  step 1: from = `ï` is replaced with to = `G`.

  step 2: accented `ó` is replaced with `o`

  step 3: `Ŧ` is a remaining non-ASCII character and replaced with _.

  In the above example, the `Ŧ` is not an “accented” character and was therefore not mapped in step 2. If the preference is not to have the `Ŧ` character mapped to the underscore character, then this character could be included in the “*<from>,<to>*” string parameter pair.

Example	Output
{{ fn.fix_non_ascii 'Têst Nón Asciïæ mapping',sæ,Ga }} {{ fn.fix_non_ascii 'Ŧest Nón Asciï' }}	'TeGt Non AGciia mapping' '_est Non Ascii'

- *fn.fix_username* : Given a string of characters, return the string with a replacement character: `-` for any character *not* in the following range: `a-zA-Z0-9._-`.

  A Unified CM Remote Destination name is an example where such a string is required.

Example	Output
{{ fn.fix_username O'Reilly }}	O-Reilly

- *fn.format_string_if_prefixed* : Given 3 input parameters:
  - a string to format
  - a string of starting characters matching the string to format
  - a prefix string

  return a string with a prefix.

  If the string of starting characters is *not* present in the string to format, no prefix is added.

Example	Output
{{ fn.format_string_if_prefixed "foobar", "foo", "AddPrefix-" }}	AddPrefix-foobar

Example	Output
{{ fn.format_string_if_prefixed "foobar", "no", "AddPrefix-" }}	foobar

- *fn.format_if_prefixed* : Given 3 input parameters:
  - a string to format *or* a list of strings

  - a string of starting characters of the string to format
  - a prefix string

return a prefixed string or the list with a prefixed string.

If the string of starting characters is *not* present in the string to format, no prefix is added.

Example	Output
`{{ fn.format_if_prefixed "foobar" "foo" "AddPrefix-" }}`	`AddPrefix-foobar`

Example	Output
`{{ fn.format_if_prefixed ["foobar","baz"] "foo" "AddPrefix-" }}`	`["AddPrefix-foobar", "baz"]`

Example	Output
`{{ fn.format_if_prefixed ["foobar","baz"] "no" "AddPrefix-" }}`	`["foobar", "baz"]`

- *fn.add_backslash_to_plus* : Given:
  - a string starting with a plus (+)

  or
  - a list of strings containing a string starting with a plus (+)

  return that string with a backslash prefix.

Example	Output
`{{ fn.add_backslash_to_plus "+123" }}`	`"\\+123"`

Example	Output
`{{ fn.add_backslash_to_plus ["+foobar", "baz"] }}`	`["\\+foobar", "baz"]`

11.3.4. List functions

- *fn.group_by_larger_than_count*: Returns either a list of instance names or dictionary instance names with count values, given parameters:
  - model type
  - model attribute
  - *greater* than an input numeric value of the attribute
  - specified search direction in hierarchy: *up* or *down*

Example	Output

Context hierarchy: `sys`, list of more than 1 of `iso_country_code` in `data/Countries`, search `down`
– display as dictionary:
`{{ fn.group_by_larger_than_count data/Countries, iso_country_code, down,1,dict }}`
– display as list:
`{{ fn.group_by_larger_than_count data/Countries, iso_country_code, down,1,list }}`
	`[ { "iso_country_code": "AUS", "count": 2 }, { "iso_country_code": "CHN", "count": 2 } ]`
	
	`[ "AUS", "CHN" ]`

- *fn.is_list*: Return true or false if the parameter is a list or not.
- *fn.list_index*: Return a specified item from a list. Zero is the first item.

Example	Output
`{{fn.list_index 2,data.Countries.country_name}}`	"Canada" if this is the third item.

- *fn.list_index_item*: Return the position of item in list.

Example	Output

`MACRO1={# fn.sequence 40,43 #}`
 	
`{{ fn.list_index_item 42,macro.MACRO1 }}`
	`2`

- *fn.list_count*: Return the number of items in a list. Note that if the list is *known* and empty, the count is 0, but if the list is *not known*, then the count is 1, because the returned message string count is 1.

Example	Output

`{{fn.list_count data.Countries}}`
 	
`{{fn.list_count input.does_not_exist}}`
 	
`{{fn.list_count non_existant_namespace.does_not_exist}}`
	`25 if the list has 25 items.`
	
	`0`
	
	`1`

- *fn.list_count_item*: Return the number of times item is in a list.

Example	Output

`MACRO1={# data.Countries.international_access _prefix #}`
 	
`{{ fn.list_index_item 00,macro.MACRO1 }}`
	`19`

- *fn.list_in*: Return true or false if item is in a list or not.

Example	Output
{{fn.list_in 'AUS',data.Countries.iso_country_code}} {{fn.list_in 'AUZ',data.Countries.iso_country_code}}	true false

- ***fn.list_contain*: Return true or false if item is a substring of an input context**
  or is in a list or not.

Example	Output

{
"input": {
"list": [
"AUS",
"BHR"
],
"key": "aaAUZa"
 	
}
}
 	
{{fn.list_contain 'AUS',input.list }}
 	
{{fn.list_contain 'AUZ',input.key }}
	true
	
	true

- *fn.list_contain_pattern*: Given a pattern or list of patterns as well as a target list of items, return the matching list of items from the target list of items.

  If an invalid pattern is input (or not matched), then it is ignored.

Example	Output

`{`
`  "input": {`
`  "pattern_list": [`
`  "SEP",`
`  "BAT"`
`  ],`
`  "desk_phones": [`
`  "SEP9999ABAB0000",`
`  "BOT123123",`
`  "TCT123123",`
`  "BAT9999ABAB0000"`
`  ]`
`  }`
`}`
	`[`
	`↪"SEP9999ABAB0000`
	`↪"]`
	
	`[`
	`↪"SEP9999ABAB0000`
	`↪",`
	`↪"BAT9999ABAB0000`
	`↪"]`

`{{fn.list_contain_pattern SEP, input.desk_phones }}`	
`{{fn.list_contain_pattern input.pattern_list, input.desk_phones }}`	

- *fn.list_append*: Returns a list with item appended.

Example	Output

`MACRO1={# fn.sequence 40,43 #}`
 	
`{{fn.list_append 999,macro.MACRO1 }}`
	`['40', '41',`
	`'42', '43',`
	`'999']`

- *fn.list_pop*: Return the last item of the list.

Example	Output

`MACRO1={# fn.sequence  40,43 #}`
 	
`{{fn.list_pop macro.MACRO1}}`
	`"43"`

- *fn.list_insert*: Return a list with item inserted at position.

Example	Output
MACRO1={# *fn.sequence 40,43* #} {{fn.list_insert 1,999,macro.MACRO1 }}	['40', '999', '41', '42', '43']

- *fn.list_insert_no_dup*: Return a list with item inserted at position and all duplicates ignored.

Example	Output
MACRO1={# *fn.sequence 40,43* #} MACRO7={# *fn.sequence 39,41* #} {{fn.list_insert_no_dup macro.MACRO1,macro.MACRO7}}	['40','41', '42','43', '39']

- *fn.list_remove*: Return a list with all instances of item or list of items removed.

Example	Output
MAC1={# *fn.sequence 40,43* #} MAC7={# *fn.sequence 40,41* #} MAC3={{fn.list_insert 1,43,macro.MAC1 }} {{fn.list_remove 43,macro.MAC3 }} {{fn.list_remove macro.MAC7,macro.MAC1 }}	['40','41', '42'] ['42','43']

- *fn.list_remove_pattern*: Given a pattern or list of patterns as well as a target list of items, remove the matching list of items from the target list of items.

  If an invalid pattern is input (or not matched), then it is ignored.

Example

```
{
    "input": {
        "pattern_list": [
            "SEP",
            "BAT"
        ],
        "desk_phones": [
            "SEP9999ABAB0000",
            "BOT123123",
            "TCT123123",
            "BAT9999ABAB0000"
        ]

    }
}
```

```
{{fn.list_remove_pattern SEP, input.desk_phones }}
```

```
{{fn.list_remove_pattern input.pattern_list, input.desk_phones }}
```

Output

```
["BOT123123",
 "TCT123123",
↪"BAT9999ABAB0000
↪"]
```

```
["BOT123123",
"TCT123123"]
```

- *fn.list_remove_dup_dict*: Given a list of dictionaries, a sort key and an optional sort direction, return a list with the duplicate entries removed, sorted by the specified sort direction or if not specified, sorted in ascending order.

Example	Output

<pre>{
  "input": {
   "dupl":
  [
    {
        "country_name": "Canada",
        "iso_country_code": "CAN"
    },
    {
        "country_name": "China",
        "iso_country_code": "CHN"
    },
    {
        "country_name": "China",
        "iso_country_code": "CHN"
    },
    {
        "country_name": "Switzerland",
        "iso_country_code": "CHE"
    }
  ]
  }
}
 	
{{ fn.list_remove_dup_dict input.dupl,
   country_name }}</pre>
	<pre>{"dupl":
	 [
	  {
	      "country_
	↪name": "Canada
	↪",
	      "iso_
	↪country_code
	↪": "CAN"
	  },
	  {
	      "country_
	↪name": "China
	↪",
	      "iso_
	↪country_code
	↪": "CHN"
	  },
	  {
	      "country_
	↪name":
	↪"Switzerland",
	      "iso_
	↪country_code
	↪": "CHE"
	  }
	]
	}</pre>

- *fn.list_remove_dup*: Return a list with all instances of item or list of items removed, including duplicates.

Example	Output
Given the following list is the result of the regex for iso_country_code:	

<pre>{# data.Countries.iso_country_code |
   iso_country_code:/AU/ #}</pre>
	<pre>['AUS','SAU',
	  'AUS','AUS']</pre>

<pre>{# fn.list_remove_dup data.Countries.iso_country_code |
   iso_country_code:/AU/ #}</pre>
	<pre>['AUS','SAU']</pre>

- *fn.list_remove_nulls*: Return a list with all `null` instances in a list of items removed.

Example	Output

{"pwf": {
"my_list": [1, null, 2,"test", 3, null]
}
}
 	
{{ fn.list_remove_nulls pwf.my_list }}
	[1,2,"test",3]

- *fn.list_reverse*: Return a list that is the reverse of a given list.

Example	Output
MACRO1={# fn.sequence 40,43 #} {{fn.list_reverse macro.MACRO1}}	['43','42', '41','40']

- *fn.list_extend*: Return a list that is an extension of list 1 with list 2.

Example	Output
MACRO1={# fn.sequence 40,43 #} MACRO9={# fn.sequence 50,52 #} {{fn.list_extend macro.MACRO1,macro.MACRO9 }}	['40', '41', '42', '43', '50', '51', '52']

- *fn.list_extend_no_dup*: Return the concatenation of list1 and list2, ignoring duplicates.

Example	Output
MACRO1={# fn.sequence 40,43 #} MACRO8={# fn.sequence 42,45 #} {{fn.list_extend_no_dup macro.MACRO1,macro.MACRO8 }}	['40','41', '42','43', '44','45']

- *fn.sequence*: Return a sequence of numbers running from the first value to the second value, optionally padded with zeroes to be the length of a third value.

Example	Output
{# fn.sequence 40,43 #} {# fn.sequence 110,100,4 #}	['40','41', '42','43'] ['0110','0109', '0108', '0107','0106', '0105','0104', '0103','0102', '0101','0100']

- *fn.list_set_symdiff*: Given two lists as input, return the list of items that are *not* common to both lists.

Example	Output

MACRO1={# fn.sequence 40,43 #}
MACRO2={# fn.sequence 41,45 #}
 	
{{ fn.list_set_symdiff macro.MACRO1,macro.MACRO2 }}
	['40','44','45']

- *fn.list_sort*: Return a sorted list; by ascending (A) or descending (D) order.

Example	Output
MACRO1={# fn.sequence 40,43 #} {{fn.list_sort macro.MACRO1,D}} {{fn.list_sort macro.MACRO1,Descending}} MACRO5={# fn.sequence 110,108,4 #} {{fn.list_sort macro.MACRO5,A}}	['43','42', '41','40'] ['43','42', '41','40'] ['0110','0109', '0108'] ['0108','0109', '0110']

- *fn.one*: Return a single result from a list. This is used to convert a single element list to a string.

  If fn.one is called with a string, it returns the string unchanged. The string can be a macro that might get the value of another attribute from a context, such as `input.some_variable`.

Example	Output
{{fn.one data.Countries.iso_country_code | emergency_access_prefix:'112'}} {{fn.one abc}}	A single result, e.g. 'DEU' 'abc'

- *fn.as_list*: Return a string result as a list. If fn.as_list is called with a string, it returns a list. Again, abc could be a macro that resolves to the value of an attribute in its context.

Example	Output
{{fn.as_list data.Countries.country_name | country_name:'China'}} {{fn.as_list abc}}	['China'] ['abc']

- *fn.list_empty*: Returns an empty list

Example	Output
{{fn.list_empty}}	[]

- *fn.list_set_intersect*: Given two lists, return the intersection as a list.

Example	Output

MACRO1={# fn.sequence 40,43 #}
MACRO2={# fn.sequence 41,42 #}
 	
{# fn.list_set_intersect macro.MACRO1,macro.MACRO2 #}
	['41','42']

- *fn.list_set_union*: Given two lists, return the union as a list.

Example	Output

MACRO1={# fn.sequence 40,43 #}
MACRO2={# fn.sequence 41,45 #}
 	
{# fn.list_set_union macro.MACRO1,macro.MACRO2 #}
	['40','41',
	'42','43',
	'44','45']

- *fn.list_set_left*: Given two lists, return a list of items in the left list only.

Example	Output

MACRO1={# fn.sequence 40,43 #}
MACRO2={# fn.sequence 41,45 #}
 	
{# fn.list_set_left macro.MACRO1,macro.MACRO2 #}
	['40']

- *fn.list_set_right*: Given two lists, return a list of items in the right list only.

Example	Output

MACRO1={# fn.sequence 40,43 #}
MACRO2={# fn.sequence 41,45 #}
 	
{# fn.list_set_right macro.MACRO1,macro.MACRO2 #}
	['44','45']

- *fn.list_filter_fields*: Given a model name, two field names, hierarchy PKID and a list of field values from one of the fields, return the list of values from the other field. The hierarchy PKID parameter must be passed in as a PKID of the hierarchy node.

Example

```
{
    "input": {
        "list": [
            "AUS",
            "BHR"
        ]
    }
}


{# fn.list_filter_fields data/Countries,
    8c0hfle2c0deab00da595101,iso_country_code,
    country_name,input.list #}
```

Output

```
[
  "Australia",
  "Bharain"
]
```

- *fn.flatten_list_of_lists*: Given a list of lists, return a single, flattened list.

Example

```
 {
     "input": {
         "args": [
             [
                 "AAA"
             ],
             [
                 "BBBB",
                 "CCCC"
             ],
             [
                 "DD"
             ]
         ]
     }
 }

{# fn.flatten_list_of_lists input.args #}
```

Output

```
[
  "AAA",
  "BBBB",
  "CCCC",
  "DD"
]
```

- *fn.flatten_nested_lists*: Given a list of scalars and lists (which could have different levels of nesting), return a single, flattened list.

Example

```
{
    "input": {
        "args": [
            "77174011",
            [
             "77174021",
             "77174080",
             [
              "55555",
              "1",
             ]
            ],
            "12345"
        ]
    }
}

{{ fn.flatten_nested_lists input.args }}
```

Output

```
[
  "77174011",
  "77174021",
  "77174080",
  "55555",
  "1",
  "12345"
]
```

- *fn.modulo_list*: Given a list and divisor, return a modulo list: list items that leave no remainder after divided. The input list is typically a directory number list and the divisor is an E164 range.

Example

```
{
    "input": {
        "args": [
            [
                "100",
                "12000",
                "13000",
            ]
    }
}

{{ fn.modulo_list input.args,1000 }}
```

Output

```
[
  "12000",
  "13000"
]
```

- *fn.get_tvpair_list*: Given an input list of dictionaries containing repeated pairs of equal type and a title-value mapping of these pairs, return a list of objects of these pairs where the pairs are mapped to `title` and `value` pairs that can for example be shown in a GUI rule in a choices drop-down list on a GUI form.

Example	Output

{
"input": {
"dataList" = [
{
"timeZoneCode": "10",
"timeZoneName": "Africa/Bissau"
},
{
"timeZoneCode": "100",
"timeZoneName": "America/Noronha"
},
...
...
]
}
}
 	
{{ fn.get_tvpair_list input.dataList,
title:timeZoneName,value:timeZoneCode }}
	[
	{
	"value": "10
	↪",
	"title":
	↪"Africa
	/
	↪Bissau"
	},
	{
	"value":
	↪"100",
	"title":
	↪"America
	/
	↪Noronha"
	},
	...
	...
	]

- *fn.list_batch*: Given a batch size and list as parameters, return the list as a batch of lists - each with size *batchsize*. If list cannot divide into `batchsize`, then the last list item is the remainder.

  The function is useful when processing large lists, which can then first be split into smaller batches.

  Example input:

```
{
  "input": {
    "list": [
      "foo",
      "bar",
      "baz",
      "qux",
      "quux"
    ]
  }
}
```

Example	Output
`input.list` as above (5-item list): `{{ fn.list_batch 2,input.list }}`	batch of 2-item lists: `[ [ "foo", "bar" ], [ "baz", "qux" ], [ "quux" ] ]`

11.3.5. Model filters

Model filters are set up as instances of `data/ModelFilterCriteria` with the following properties:

Property	Description
`name`	The name is used by the macro function
`type`	The model on which the filter is applied
`criteria.attribute`	attribute name of the applied model
`criteria.conditional_operator`	optional: takes a boolean AND, OR NOT to apply to next criteria
`criteria.condition`	see list below for options
`criteria.value`	filter condition value of attribute to check against

`criteria.condition` can be:

Field Name	Description	GUI Field Name
`equalsIgnoreCase`	Case insensitive exact match.	**Equals**
`isexactly`	Case sensitive exact match.	**Equals Exactly**
`containsIgnoreCase`	Case insensitive substring match.	**Contains**
`contains`	Case sensitive substring match.	**Contains Exactly**
`regex_search`	Regular expression	**Regex Search**

The macro functions take a Model Filter name as input.

Match model filter criteria

- *fn.match_model_filter_criteria*: Given input context data and a model filter criteria instance `name`, return a dictionary of output that contains a `result`.

  Syntax: *{{fn.match_model_filter_criteria <input-data>,<filter-name> }}*

  In the Model Filter Criteria:

  - The `filter-name`, individual `criteria` are evaluated to a value: `True` or `False`
  - The `filter-name`, criteria are combined with a `conditional_operator` (e.g. boolean `AND`) to yield the `result` value: `true` or `false`
  - Filters will evaluate each of the criteria for the specified `type`.

  Below is an instance of `data/ModelFilterCriteria` for `type` is `device/msgraph/MsolUser`:

```
"data": {
           "type": "device/msgraph/MsolUser",
           "description": "Multi Level Filter",
           "name": "Multi Level Filter",
           "criteria": [
               {
                   "attribute": "City",
                   "condition": "isexactly",
                   "value": "Cape Town",
                   "conditional_operator": "AND"
               },
               {
                   "attribute": "Address",
                   "condition": "contains",
                   "value": "Bellville"
               }
            ]
        }
```

Example	Output

`data/ModelFilterCriteria` with `name` is "Multi Level Filter" as in example above.
Input context data contains:
`{"user_details": { "City": "Cape Town", ... "Address": "City of Bellville", ... }`
Function call:
`{{fn.match_model_filter_criteria user_details.City, 'Multi Level Filter' }}`
	`{"output": "[City] 'Cape Town' isexactly 'Cape Town' = True\n [Address] 'City of Bellville' contains 'Bellville' = True", "result": true}`

Instances that match model filter criteria

- *fn.instances_match_model_filter_criteria*:

  Syntax:

  {{fn.instances_match_model_filter_criteria <model filter criteria name>,<field name>,<direction>}}

  Given:

  - model filter criteria name: `name` of `data/ModelFilterCriteria`
  - field name: a field name from the model filter criteria instance `type`. The match results will be a list of this field, with the list called `model_match_list`.

    The model filter can contain one or multiple `criteria` operating on the `type` model - with multiple criteria also combined with a boolean `conditional_operator`.
  - direction: optional search direction. Default is `above`, which is upwards and excludes the current hierarchy if it is not the top hierarchy.

  Return:

  - Input model filter and query details and result `model_match_list`.

  Example macro:

```
{{ fn.instances_match_model_filter_criteria CityTulsa,username,up }}
```

  Example output:

```
{
  "model_filter_criteria_field": "username",
  "model_filter_criteria_name": "CityTulsa",
  "hierarchy": "nnf525b7e04a4a001480cfnn",
```

(continues on next page)

(continued from previous page)


```
  "model_filter_criteria": {
      "usage": "Test",
      "type": "data/User",
      "name": "Demo Criteria City = Tulsa",
      "criteria": [
          {
              "attribute": "city",
              "condition": "isexactly",
              "value": "Tulsa"
          }
      ]
  },
  "model_filter_macro": "{# data.User.* || direction:up #}",
  "model_match_list": [
      "IsaiahL@tobuild.onsoft.com",
      "PattiF@tobuild.onsoft.com",
      "LidiaH@tobuild.onsoft.com",
      "pietman@tobuild.onsoft.com",
      "HenriettaM@tobuild.onsoft.com"
  ],
  "model_filter_criteria_direction": "up",
  "hierarchy_friendly_path": "sys.hcs.CS-P.CS-NB.RND.East.AlterLake"
}
```


Build a macro for filtering drop-down choices

fn.build_filter_macro: Given 2 mandatory input parameters:

- a
- a

return

Examples:

Example	Output

11.3.6. Model instance filters from CSV file

fn.generate_filters: given input parameters:

- An uploaded CSV filename: the filename of a file uploaded to `data/File`, e.g. using the **File Management** menu.
- Model Instance Filter (MIF) name: a name for an instance of `data/ModelInstanceFilter`
- A model type to apply the MIF to, e.g.: `device/msteamsonline/CsOnlineUser`.
- A filter type: `inclusion` or `exclusion` (if not provided as parameter, default is `inclusion`)

create a MIF (`data/ModelInstanceFilter`) instance at the hierarchy where the function is run, containing the data in the CSV file. The MIF can then be selected in data syncs at the relevant hierarchy.

Syntax: *{{ fn.generate_filters <InputFile.csv>,<Output_MIF_Name>,<input_model>[,<filter_type>] }}*

Note: The MIF `model_parent` will be the parent of the input model. For example, if the input model is `device/msteamsonline/CsOnlineUser` then `device/msteamsonline` is the `model_parent`.

The CSV file format is interpreted as follows:

- Header row: attribute(s) of *<input_model>* (`attr_name`)
- 2nd row onwards: value(s) which have `equals` match to attribute(s)

Note:

- Each value row in the CSV file represents a model filter instance with value `equals` the `attr_name`. Filter instances are combined with logical OR. Refer to the *Model Instance Filter* topic in the Core Feature Guide.
- If the CSV file contains more than one column, i.e multiple attributes and values, then the `attr_filters` list in the MIF instance will combine these with a logical AND. Refer to the *Model Instance Filter* topic in the Core Feature Guide.

Example

For example, a CSV file (`InputFile.csv`) uploaded using the **File Management** menu at hierarchy `sys.hcs.CS-P.VOSS-OPS.VOSS` with a single attribute `UserPrincipalName`:

```
UserPrincipalName
Dusty.Moyer@visionoss-dev.info
Earl.Moore@visionoss-dev.info
```

then the macro function run at hierarchy `sys.hcs.CS-P.VOSS-OPS.VOSS` called

```
{{ fn.generate_filters InputFile.csv,MSOL_MIF,device/msteamsonline/CsOnlineUser }}
```

will create MIF instance containing:

```
{
 "meta": {},
 "resources": [ {
    "meta": {
      "model_type": "data/ModelInstanceFilter",
      "pkid": "...",
      "schema_version": "...",
      "hierarchy": "sys.hcs.CS-P.VOSS-OPS.VOSS",
      "tags": []
    },
    "data": {
      "filter_type": "inclusion",
      "name": "MSOL_MIF",
      "model_parent": "device/msteamsonline",
```

(continues on next page)

(continued from previous page)

```
      "model_filters": [
        { "model_type": "device/msteamsonline/CsOnlineUser",
         "attr_filters": [
           { "attr_name": "UserPrincipalName",
            "condition": "equals",
            "value": "Dusty.Moyer@visionoss-dev.info"
           } ] },
        { "model_type": "device/msteamsonline/CsOnlineUser",
         "attr_filters": [
           { "attr_name": "UserPrincipalName",
            "condition": "equals",
            "value": "Earl.Moore@visionoss-dev.info"
           } ] }
      ] }
   } ]
}
```

11.3.7. Rule filter functions

The “filter by rule” function returns a list of resource instance data for a given model type. The schema of the model type in question must define a `rules` object of the form:

```
{
   'rules': {
          'hierarchy_types': [<list of data/HierarchyNodeType business keys>]
    }
}
```

The model `data/Role` is one example of such a model type. Filtering is applied using the current hierarchy context or else based on an explicit hierarchy type name.

Macro format:

```
{{ fn.filter_by_rule <rule name>,
                     <model type>,
                     <direction>,
                     <attribute path to return>,
                     <hierarchy type name> }}
```

Argument descriptions:

<rule name>:
: The name of the rule being used to filter results. Supported values: `hierarchy_types`

<model type>:
: The model type of the instances to be filtered.

<direction>:
: The search direction of the results. Possible values are:

  - `all` - search at current hierarchy, ancestors, and descendants.
  - `up` - search at current hierarchy and ancestors.

- `down` - search at current hierarchy and descendants.
- `local` - search at current hierarchy only.

<attribute path to return>: [optional]
The path dot (.) delimited path to a single attribute to return.

- If not specified the returned result will contain a list of objects.
- If specified the returned result will contain a list the given attribute.
- Must be “null” if this field is not required, while <hierarchy type name> is supplied.

<hierarchy type name>: [optional]
The name of the hierarchy type to filter by. This will be looked up from the current hierarchy going up.

Examples:

```
{{ fn.filter_by_rule hierarchy_types,data/Role,up,name,Customer }}
```

Returns all the names of the roles that are permitted at “Customer” hierarchy type. Lookup is done from the current hierarchy upwards.

```
{{ fn.filter_by_rule hierarchy_types,data/Role,up,null,Customer }}
```

Returns full instance data of the roles that are permitted at “Customer” hierarchy type. Lookup is done from the current hierarchy upwards.

```
{{ fn.filter_by_rule hierarchy_types,data/Role,up,name }}
```

Returns all the names of the roles that are permitted at the hierarchy type of the current hierarchy context. Lookup is done from the current hierarchy upwards.

```
{{ fn.filter_by_rule hierarchy_types,data/Role,up }}
```

Returns full instance data of the roles that are permitted at the hierarchy type of the current hierarchy context. Lookup is done from the current hierarchy upwards.

11.3.8. Role at allowed hierarchy function

Given a hierarchy friendly path as a function parameter, the `fn.get_admin_roles_allowed_at_hn` function returns the list the roles allowed for admins at the given hierarchy friendly path.

For example, given a context hierarchy (site):

```
sys.hcs.CS-P.CS-NB.AAAGlobal.LOC001
```

and evaluating the function:

```
{{ fn.get_admin_roles_allowed_at_hn fn.hierarchy_friendly_path }}
```

the output of this macro will for example be:

```
[
    "LOC001SiteAdmin",
    "LOC001SiteOper"
]
```

Given a context hierarchy (reseller):

```
sys.hcs.CS-P.CS-NB
```

the output of this macro will for example be:

```
[
    "CS-NBCustomerAdministrator",
    "CS-NBCustomerOperator",
    "CS-NBResellerAdministrator",
    "CS-NBResellerOperator",
    "CS-NBSiteAdmin",
    "CS-NBSiteOper"
]
```

11.3.9. Filter role functions

Given a list of user roles as a function parameter, the `fn.filter_roles_by_user_access_profile` function returns the list of roles with permissions equal to or less than that of the user executing the function.

For example, given a list of roles:

```
macro.ALL_Roles

[TestRole01, TestRole02, TestRole03]
```

where `TestRole01` has the least permissions and `TestRole03` has the most permissions. If the request user is assigned with `TestRole02`, and executes the following macro:

```
{{ fn.filter_roles_by_user_access_profile macro.ALL_Roles }}
```

The output of this macro will be

```
[TestRole01, TestRole02]
```

Note: This filter will only have an effect if the setting in `data/Settings` called Additional Role Access Profile Validation is enabled (check box is enabled).

Refer to the *Additional role access profile validation* settings topic.

11.3.10. Macro evaluate function

- *fn.evaluate*: Evaluate the string using the macro interpreter.

  The string can be a macro and can also contain macro names to be evaluated.

  The purpose of the function is so that we can save data as a macro and then evaluate it when we read it again.

  For example, we may store default values in a data model (e.g. `SiteDefaultsDoc`) that may also refer to existing macros. An attribute of the model - with name: `defaulthppt` can for example have the value (The macro evaluates to a site name):

  ```
  Site-{{macro.SITENAME}}
  ```

  Then, a workflow Configuration Template at a Site can be a set to a value that references the model and that uses `fn.evaluate`:

  ```
  "defaulthppt": "{{ fn.evaluate data.SiteDefaultsDoc.defaulthppt }}",
  ```

  Not:

  ```
  "defaulthppt": "Site-{{macro.SITENAME}}",
  ```

The tables below provide further examples.

Where `self.x` specifies an attribute of a model, with the value:

```
{# data.Countries.iso_country_code | country_name:'South Africa' #}
```

Example	Output
Then: `{# fn.evaluate self.x #}`	`['ZAF']`

Where `MACRO1` is:

```
{{ data.Countries.emergency_access_prefix | iso_country_code:FRA }}
```

Where self.x is:

```
{# data.Countries.iso_country_code | emergency_access_prefix:macro.MACRO1 #}
```

Example	Output
Then: `{# fn.evaluate self.x #}`	Codes with same prefix as FRA: `['DNK','HKG', 'FRA','DEU', 'IND','ITA', 'NLD','NZL', 'SAU','ESP', 'SWE','ZAF', 'CHE','TUR']`

11.3.11. CUCM and device functions

- *fn.get_cucms_associated_via_ndlr*: Get all CUCM servers from NDLR (Network Device List Reference) on current the site. The function must be called from a site hierarchy.

Example	Output
hierarchy: `sys.hcs.CS-P.CS-NB.Geologic.EMEA.Paarl` `{# fn.get_cucms_associated_via_ndlr #}`	`[ 10.110.11.131 ]`

- *fn.get_cucm_bkeys_associated_via_ndl*: Get all CUCM server business keys from NDL (Network Device List) on current the hierarchy. A business key is a list of: IP, port, hierarchy.

Example	Output
hierarchy: `sys.hcs.CS-P` `{# fn.get_cucm_bkeys_associated_via_ndl #}`	`[ "[\"10.110.11.131\", \"8443\", \"hcs.CS-P.CS-NB.AAAGlobal\"]" ]`

- *fn.cucm_get_line_details*: Specify the line pattern and routePartitionName and use the Macro Evaluator function to view the line parameters for the specified line.

  To return the result for a single line parameter, append the required parameter to the end of the macro function, for example:

  `{{ fn.cucm_get_line_details 4025 }}`

  The macro can only be run at or below the hierarchy level of the Unified CM that is provisioned.

  If no routePartitionName is specified, then only the line directory number pattern is searched for on the Unified CM:

```
{{ fn.cucm_get_line_details 4025, }}
```

In a multi-cluster environment with more than one device, then an additional optional parameter should be used to specify the Unified CM - its pkid. In this case, using the device parameter also requires that the `all` parameter be used if all parameters need to be returned.

Multi cluster example (for `data/CallManager/57e709467677f0c9ca956f6f`)

Example	Output
`{{ fn.cucm_get_line_details 4025,VS-Corp-NewYork, all,57e709467677f0c9ca956f6f}}`	`{ "is_line_shared": true, "remote_destination_profiles": [ "RDP_vdevenr" ], "device_profiles": [ "UDP_vdevenr" ], "phones": [ "SEP002155D547F7", "SEP111122223333" ] }`

Single cluster examples:

Example	Output
`{{ fn.cucm_get_line_details 4025,VS-Corp-NewYork }}`	`{ "is_line_shared": true, "remote_destination_profiles": [ "RDP_vdevenr" ], "device_profiles": [ "UDP_vdevenr" ], "phones": [ "SEP002155D547F7", "SEP111122223333" ] }`
`{{ fn.cucm_get_line_details 4025,VS-Corp-NewYork, is_line_shared }}`	`true`
`{{ fn.cucm_get_line_details 4025,VS-Corp-NewYork, phones }}`	`SEP002155D547F7 SEP111122223333`

- *fn.get_endpoint_name*

  Given a set of gateway input parameters, return an endpoint name. The input parameters are ordered:

  1. gateway product
  2. gateway protocol
  3. gateway name
  4. gateway module
  5. gateway slot (int)
  6. gateway subunit
  7. gateway subunit position (int)
  8. gateway endpoint port (int)
  9. gateway endpoint product

```
{{ fn.get_endpoint_name
    VG350 SCCP SKIGW1122111111␣
↪ANALOG
    2 SM-D-48FXS-E-SCCP 0 46
↪'Analog Phone' }}
```

```
AN112211111142E
```

```
{{ fn.get_endpoint_name
    VG350 MGCP test1.com ANALOG
    2 SM-D-72FXS 0 9 'Cisco MGCP␣
↪FXS Port' }}
```

```
AALN/S2/SU0/9@test1.com
```

- *fn.get_sccp_endpoint_name*

  Remains available as an alias for *get_endpoint_name* to support existing features. Given a set of gateway input parameters, return an endpoint name. The input parameters are ordered:

  1. gateway product
  2. gateway protocol
  3. gateway name
  4. gateway module
  5. gateway slot (int)
  6. gateway subunit
  7. gateway subunit position (int)
  8. gateway endpoint port (int)
  9. gateway endpoint product

```
{{ fn.get_sccp_endpoint_name
    VG350 SCCP SKIGW1122111111 ANALOG
    2 SM-D-48FXS-E-SCCP 0 46 'Analog␣
↪Phone' }}
```

```
AN112211111142E
```

- *fn.lines_from_hierarchy_devices*

  Given a hierarchy and device (one of Phone, Device Profile or Remote Destination Profile), return the list of patterns and routePartitionNames for the device at the hierarchy.

  Example:

```
{{ fn.lines_from_hierarchy_devices
   sys.hcs.CS-P.OBCust.OBSite1
   Phone }
```

```
{ "pattern":  ["555555","710087"],
  "routePartitionName": [None,"Site-CPT1
↪"]}
}
```

- *fn.default_device*

  Given a hierarchy and device (or device and path to pkid), return the default device at the hierarchy according to the device in the Network Device List (NDL) of the hierarchy. Calling the function without a pkid path returns the entire object. If no default device is found, an empty string is returned.

  Example:

```
{{ fn.default_device data/CallManager.
↪pkid }}


{{ fn.default_device data/CallManager }}
```

```
69cer80r903aa8b565784675


[
   {
     "pkid": "59ccdc58dcdbfaa51eedbed9",
     "model_type": "data/CallManager",
     "uri": "/api/v0/data/CallManager/
↪69cer80r903aa8b565784675
   }
]
```

- *fn.device_meta*

  Can take 0, 1 or 2 parameters to return NDL data.

  - If no parameters, the function should be called at a site hierarchy. Returns NDL object data (e.g. details of `data/CallManager`, `data/UnityConnection`)
  - If 1 parameter, it must be the hierarchy: as friendly name or `fn.hierarchy`.
  - If 2 parameters, hierarchy followed by comma, then object name and optionally dot and attribute (pkid example below)

  Example:

{{ fn.ndl_device_meta fn.hierarchy, ndl.data/CallManager.pkid }}	69cer80r903aa8b565784675
{{ fn.ndl_device_meta fn.hierarchy }}	{ "ndl": { "name": "NDL-GeoLogic-1", "pkid": "59ccdcd5303aa8b5657426ac", "data/CallManager": { "pkid": "69cer80r903aa8b565784675", "bkey": "[\"10.140.51.164\", \"8443\", \"hcs.CS-P.CS-NB.GeoLogic\"]" }, "bkey": "[\"NDL-GeoLogic-1\", \"hcs.CS-P.CS-NB.GeoLogic\"]", "data/Hcmf": { "pkid": "59ccd953dc66faa51eeda8d5", "bkey": "[\"10.140.51.156\", \"8443\", \"hcs\"]" }, "data/UnityConnection": { "pkid": "59ccdc8cb969f577e64fcc98", "bkey": "[\"10.140.51.165\", \"8443\", \"hcs.CS-P.CS-NB.GeoLogic\"]" } } }

- *fn.get_least_used_site_devicepool*

Important: This function can only be run at a site and only searches locally, so no device selection is required.

Return the CUCM DevicePool with the lowest phone count at the submitted site.

Example:

{{ fn.get_least_used_site_devicepool }}	Cu1Si1-CAT-Dallas-SRST-DP1

- *fn.generic_device_model_custom_operation*

  Used to get return values on custom operations for generic driver device models for use in a workflow context.

  Given parameters:

  - full generic driver device type, e.g. `device/genericcucm/PbrMac`
  - operation_name: e.g. `get_mac`
  - input_data: as dictionary, e.g. `{"extension": "1094"}`

  return the value of the custom operation as a dictionary

  Example:

```
{{ fn.generic_device_model_custom_
↪operation
   device/genericcucm/PbrMac
   get_mac
   \{"extension": "1094"\}
}}
```

```
{"MAC": "001122AABBCC"}
```

11.3.12. Jabber device name function

- *fn.jabber_device_name*: Given a Jabber device type string and a username, return a unique Jabber device name with prefix of the associated type. The maximum allowed total length is 15 characters.

  Jabber device type strings and automatic prefixes:

  - 'Cisco Dual Mode for Android': 'BOT'
  - 'Cisco Jabber for Tablet': 'TAB'
  - 'Cisco Dual Mode for iPhone': 'TCT'
  - 'Cisco Unified Client Services Framework': 'CSF'

  The function generates the device name in the following format:

  <device type prefix><username hash><random number>

  The total length of the generated devicename is 14 characters.

Note:

- Non-alphanumeric characters in usernames are replaced by zeros (0). Letters are capitalized.
- If the username is longer than 8 characters, it is truncated to 8 characters. Then the random number has a length of 3 numbers.

  Example with username "D'Malleybazfrobbleverylongname": `TCTD0MALLEY218`
- If the username is shorter than 8 characters, all characters are used. The random number has a length to make up a total of 11 characters.

  Example with username 'FOO': `TCTF0000183752`

- The generated name is checked against existing device names and the random number is regenerated until the name is unique.

Example	Output
{{ fn.jabber_device_name 'Cisco Dual Mode for iPhone', D'Malleybazfrobbleverylongname }}	TCTD0MALLEY218

11.3.13. Webex App functions

fn.get_webex_teams_user_csv_data_specific_user

Given a user's e-mail address as input, return the full comma-separated value (CSV) data for a specific Webex App user.

Input:

```
{{ fn.get_webex_teams_user_csv_data_specific_user user1@email.com }}
```

Returns:

```
User ID/Email (Required),Jabber with Webex App,Jabber Calling,UC Manager Profile,Contact␣
↪Migration Required,Calling Behavior,Care Digital Channel,
user1@email.com,TRUE,TRUE,Profile 1,TRUE,CALL_WITH_APP_REGISTERED_FOR_CISCOTEL,TRUE
```

fn.get_webex_teams_user_csv_data_all_users

Return CSV data for all Webex App users.

Input

```
{{ fn.get_webex_teams_user_csv_data_all_users }}
```

```
User ID/Email (Required),Jabber with Webex App,Jabber Calling,UC Manager Profile,Contact␣
↪Migration Required,Calling Behavior,Care Digital Channel
user1@email.com,TRUE,TRUE,Profile 1,TRUE,CALL_WITH_APP_REGISTERED_FOR_CISCOTEL,TRUE
user2@email.com,TRUE,TRUE,,TRUE,CALL_WITH_APP_REGISTERED_FOR_CISCOTEL,TRUE
```

fn.send_webex_teams_message

Given a user's e-mail address, This function sends a Webex App message to a single recipient (person). The function uses a markdown template (configured in VOSS) for the message body. This is a Jinja template and supports pwf variables. It optionally attaches a specified file.

```
{{ fn.send_webex_teams_message username@company.com,
   pwf.wt_message_template,
   pwf.csv_file_path }}
```

fn.send_webex_teams_message_email_group

This function sends a Webex App message to multiple recipients defined in an email group i.e. data/EmailGroup Uses a markdown template (configured in VOSS) for the message body. This is a Jinja template and supports pwf variables. Optionally attaches a specified file

```
{{ fn.send_webex_teams_message_email_group pwf.email_group,
   pwf.wt_message_template,
   pwf.csv_file_path }}
```

fn.get_webex_teams_device_activation_code

Get an activation code for a Webex App device using the Place name as input. This is done by invoking the model operation directly.

```
{{ fn.get_webex_teams_device_activation_code Boardroom }}
```

11.3.14. Subscriber functions

- *fn.process_subscriber_line_data*: Used in workflows - a single parameter called input is the workflow input context, containing line data. The function returns line patterns and partitions found in any of Phone, DeviceProfile, RemoteDestinationProfile.

Examples:

Example	Output
`{{ fn.process_subscriber_line_data input }}`	`[{ "pattern": "10003", "routePartitionName": "Site-23m-Customer 1 Site A" },{ "pattern": "10005", "routePartitionName": "Site-23m-Customer 1 Site A" }]`

11.3.15. Quick add group functions

- *fn.get_qag_choices*: Used to filter Quick Add Groups (QAGs).

  0, 1 or 2 parameters are available

  0: Return all QAGs up and down from the global setting lookup level (highest level configuration: Provider). Refer to the entry for **Quick Add Group & User Profile lookup level** in the Global Settings topic of the Core Feature Guide.

  1: specify the vendor to filter QAG's by (valid options: `cisco`, `webexapp`, `microsoft`) This parameter is applied for Quick Add Group selection in the portal for the following:

  – Cisco Quick User

- WebexApp Quick User
- Microsoft Quick User

Note:

- Cisco Quick Add User can be used for adding Cisco UCM-based services, including Webex, where the user requires UCM calling.
- Webex Quick User can be used for adding Webex services where Cisco UCM calling is not required. This may be free calling or Webex Calling services.

2: integer value to specify the number of QAGs to return

Examples:

Example	Output
`{{ fn.get_qag_choices }}`	`[{ "title": "IMS-integrated Mobile (Basic)", "value": "IMS-integrated Mobile (Basic)" }, { "title": "default", "value": "default" }, { "title": "Generic Single Screen Room System", "value": "Generic Single Screen Room System" }, ...`
`{{ fn.get_qag_choices cisco }}`	`[{ "title": "Cisco DX80 Phone Type", "value": "Cisco DX80 Phone Type" }, { "title": "Cisco E20 Phone Type", "value": "Cisco E20 Phone Type" }, { "title": "Cisco 3905 Phone Type", "value": "Cisco 3905 Phone Type" }, ...`
`{{ fn.get_qag_choices cisco, 1 }}`	`[{ "title": "Cisco DX80 Phone Type", "value": "Cisco DX80 Phone Type" }]`

11.3.16. Zero, unset, boolean, drop, null, and exists functions

Function	Description	Example	Output or Result
fn.zero	Return a zero value.	`{{fn.zero}}`	`0`
fn.unset	Return an empty string.	`{{fn.unset}}`	`''`
fn.true	Return a boolean True.	`{{fn.true}}`	`true`
fn.false	Return a boolean False.	`{{fn.false}}`	`false`

Given input context:

1. `{"input": {"field1": {"key1": "value1"}}}`
2. `{"input": {"field1": None}}`
3. `{"input": {"field1": ""}}`

Function	Description	Example	Input and Result
fn.is_none_or_empty	Return a boolean if the argument is None or an empty string	`{{fn.is_none_or_empty input.field1}}`	input 1. `false` input 2. `true` input 3. `true`

Note: If `fn.is_none_or_empty` is used along with inline macro queries that have where/option clauses, the macro should be split into 2 parts.

Examples:

1. Instead of:

```
(( fn.is_non_or_empty data.User.username | username:input.username == fn.true )) <> <>
```

write as follows:

```
  USERNAME= {{ data.User.username | username:input.username }}

  (( fn.is_non_or_empty pwf.USERNAME == fn.true )) <> <>


2. Instead of:

::

   { "set_var_name": "user_role",
     "set_var_value": "(( fn.is_none_or_empty data.User.role | username:pwf.username |␣
↪direction:down == false ))<{{ data.User.role | username:pwf.username || direction:down␣
↪}}><{{ macro.LOCAL_SELFSERVICE_ROLE }}>"
   },

write as follows:

::

    { "set_var_name": "data_user_role",
      "set_var_value": "{{ data.User.role | username:pwf.username | direction:down }}"
    },
    { "set_var_name": "user_role",
      "set_var_value": "(( fn.is_none_or_empty pwf.data_user_role == false ))<{{ pwf.
↪data_user_role }}><{{ macro.LOCAL_SELFSERVICE_ROLE }}>"
    },
```

Given input context:

```
{
 "input": {
   "test": null
 }
}
```

Function	Description	Example	Output or Result
fn.null	Returns Null. Useful in comparison tests.	`(( input.test == fn.null ))`	`true`

Function	Description	Example	Output or Result
fn.drop	Removes the attribute from CFT	`{'name': '{{fn.drop}}'}`	Attribute as in existing data
fn.force_null	Attribute has Null value	`{'name': '{{fn.force_null}}'}`	`{'name': None}`

Useful in Configuration Templates (CFT):

Function `fn.drop` in a CFT will drop a value from the input data when it is processed through a CFT. Input data without the dropped field is then overlaid onto the existing model data by the workflow code. The final payload data for the value will thus *remain the same* as in the existing model data.

Example of `fn.drop` in an if-then-else test from a CFT of a workflow:

```
"ldapDirectoryName": "(( input.cucm_user_ldapDirectoryName != '' ))
  <{{ input.cucm_user_ldapDirectoryName }}>
  <{{ fn.drop }}"
```

Function	Description	Example	Output or Result
fn.drop_from_payload	Removes the attribute from payload	`{'name': '{{fn.drop_from_payload}}'}`	Attribute “blank”

Useful in Configuration Templates (CFT):

Function *fn.drop_from_payload* forces a field to not be present in the final payload data. Used in a CFT, it will drop a value from the existing model data *after* the input data is overlaid onto the existing model data. The value is then not present, i.e. “blanked”.

Example of `fn.drop_from_payload` in a CFT so that attributes are blanked:

```
{
  "meta": {},
  "resources": [
    {
      "data": {
        "name": "ConvertCUCUserCFT",
        "target_model_type": "device/cuc/User",
        "template": {
          "LdapType": "0",
          "LdapCcmUserId": "{{ fn.drop_from_payload }}",
          "LdapCcmPkid": "{{ fn.drop_from_payload }}"
        }
```

(continues on next page)

(continued from previous page)

```
      },
      "meta": {
        "model_type": "data/ConfigurationTemplate",
        "system_resource": true,
        "tags": [
          "convertusertype"
        ]
      }
    }
  ]
}
```

Given input context:

1. `{"input": {"field1": {"key1": "value1"}}}`
2. `{"input": {"field1": {"key1": null}}}`
3. `{"input": {"field1": ""}}`

Function	Description	Example	Input and Result

fn.exists	Return a boolean true if the argument is exits and has a non-null value.	`{{ fn.exists input.field1.key1 }}`	input 1.
			`true`
			input 2.
			`false`
			input 3.
			`false`

fn.exists	If the *argument does not exist*, it is interpreted as a string - which is interpretable, so the function returns true.	`{{ fn.exists field.key }}`	input 1, 2, 3 `true`

11.3.17. Time functions

- *fn.now*: Return the date and time at this moment. An optional format parameter is available.

  Example:

Example	Output
{{fn.now}}	2013-04-18 10:50:52.105130
{{fn.now "%Y%m%d"}}	20140327
macro.DAY="%A %m/%d/%Y" {{fn.now macro.DAY}}	"Thursday 03/27/2014"

  Supported date and time formats:

%a	abbreviated weekday name according to the current locale
%A	full weekday name according to the current locale
%b	abbreviated month name according to the current locale
%B	full month name according to the current locale
%c	preferred date and time representation for the current locale
%C	century number (the year divided by 100 and truncated to an integer, range 00 to 99)
%d	day of the month as a decimal number (range 01 to 31)
%D	same as m/d/y
%e	day of the month as a decimal number, a single digit is preceded by a space (range ‘ 1’ to ‘31’)
%g	like G, but without the century
%G	The 4-digit year corresponding to the ISO week number
%h	same as b
%H	hour as a decimal number using a 24-hour clock (range 00 to 23)
%I	hour as a decimal number using a 12-hour clock (range 01 to 12)
%j	day of the year as a decimal number (range 001 to 366)
%m	month as a decimal number (range 01 to 12)
%M	minute as a decimal number
%n	newline character
%p	either ‘AM’ or ‘PM’ according to the given time value, or the corresponding strings for the current locale
%P	like p, but lower case
%r	time in a.m. and p.m. notation equal to I:M:S p

continues on next page

Table 1 – continued from previous page

%R	time in 24 hour notation equal to H:M
%S	second as a decimal number
%t	tab character
%T	current time, equal to H:M:S
%u	weekday as a decimal number [1,7], with 1 representing Monday
%U	week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
%V	The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week.
%w	day of the week as a decimal, Sunday being 0
%W	week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
%x	preferred date representation for the current locale without the time
%X	preferred time representation for the current locale without the date
%y	year as a decimal number without a century (range 00 to 99)
%Y	year as a decimal number including the century
%z	numerical time zone representation
%Z	time zone name or abbreviation
%%	a literal '%' character

- *fn.now_in_tz*:

  Optional parameters:

  – Timezone, entered as listed in *Timezones*.

  – time format parameter - the time formatting is as with the Supported date and time formats table at *fn.now*.

  Without parameters: output is the same as *fn.now*.

  Example:

Example	Output
If current time is `2020-07-01 08:05:07.454918`, then	
`{{ fn.now_in_tz Australia/Melbourne,%Y-%m-%d %H:%M:%S }}`	`2020-07-01 18:08:21`
`{{ fn.now_in_tz US/Hawaii }}`	`2020-06-30 22:09:01.935328-10:00`
`{{ fn.now_in_tz Etc/GMT }}`	`2020-07-01 08:09:28.717503+00:00`
`{{ fn.now_in_tz }}`	`2020-07-01 08:09:52.413320`

- *fn.seconds_to_text*: Given an integer seconds value, convert to days, hours, minutes, seconds

Examples	Output
`{{ fn.seconds_to_text 345435 }}`	`3 days, 23 hours, 57 minutes, 15 seconds`
`{{ fn.seconds_to_text 8000 }}`	`2 hours, 13 minutes, 20 seconds`
`{{ fn.seconds_to_text 35 }}`	`35 seconds`

- *fn.add_x_days_to_startdate*: Given three arguments,
  a. integer number of days (positive or negative value)
  b. start date
  c. date-time format specification

  return a date in the past or future.

  The arguments can be named macros.

Examples	Output
{{ fn.add_x_days_to_startdate 10, '2019-10-01', '%Y:%m:%d:%I:%M' }}	2019:10:11:02:09
macro.global_setting_cooling_duration = 10 macro.TODAY_YYYY_MM_DD = '2019-10-01' macro.DateTimeFormatter_YYYY_MM_DD = '%Y:%m:%d:%I:%M' {{ fn.add_x_days_to_startdate macro.global_setting_cooling_duration, macro.TODAY_YYYY_MM_DD, macro.DateTimeFormatter_YYYY_MM_DD }}	2019:10:11:02:09

- *fn.compare_timestamps*: Given two timestamp arguments `ts1` and `ts2` and also 2 timestamp formats `tf1` and `tf2`, return a value:
  - 1 if `ts1` > `ts2`
  - 0 if `ts1` equals `ts2`
  - -1 if `ts1` < `ts2`

  The timestamp format follows the Python `datetime` library standard (`strftime(format)` method, for example `tmstamp.strftime(%Y-%m-%dT%H:%M:%S)`).

Note: The comma-separated list of parameters should not have spaces between them.

Examples	Output
`{{ fn.compare_timestamps  2019-10-01, 2020-10-01, %Y-%m-%d, %Y-%m-%d }}`	`-1`
Comparison of all values in format `{{ fn.compare_timestamps 2020-02-13T20:45:01, 2020-02-13T20:45:02, %Y-%m-%dT%H:%M:%S, %Y-%m-%dT%H:%M:%S }}`	`-1`
No comparison of omitted values in format `{{ fn.compare_timestamps 2020-02-13T20:45:01, 2020-02-13T20:45:02, %Y-%m-%d, %Y-%m-%d }}`	`0`
`{{ fn.compare_timestamps pwf.timestamp_now, pwf.timestamp_expiry, %Y-%m-%d, %Y-%m-%d }}`	`1`

Timezones

Valid timezones list is [‘Africa/Abidjan’, ‘Africa/Accra’, ‘Africa/Addis_Ababa’, ‘Africa/Algiers’, ‘Africa/Asmara’, ‘Africa/Asmera’, ‘Africa/Bamako’, ‘Africa/Bangui’, ‘Africa/Banjul’, ‘Africa/Bissau’, ‘Africa/Blantyre’, ‘Africa/Brazzaville’, ‘Africa/Bujumbura’, ‘Africa/Cairo’, ‘Africa/Casablanca’, ‘Africa/Ceuta’, ‘Africa/Conakry’, ‘Africa/Dakar’, ‘Africa/Dar_es_Salaam’, ‘Africa/Djibouti’, ‘Africa/Douala’, ‘Africa/El_Aaiun’, ‘Africa/Freetown’, ‘Africa/Gaborone’, ‘Africa/Harare’, ‘Africa/Johannesburg’, ‘Africa/Juba’, ‘Africa/Kampala’, ‘Africa/Khartoum’, ‘Africa/Kigali’, ‘Africa/Kinshasa’, ‘Africa/Lagos’, ‘Africa/Libreville’, ‘Africa/Lome’, ‘Africa/Luanda’, ‘Africa/Lubumbashi’, ‘Africa/Lusaka’, ‘Africa/Malabo’, ‘Africa/Maputo’, ‘Africa/Maseru’, ‘Africa/Mbabane’, ‘Africa/Mogadishu’, ‘Africa/Monrovia’, ‘Africa/Nairobi’, ‘Africa/Ndjamena’, ‘Africa/Niamey’, ‘Africa/Nouakchott’, ‘Africa/Ouagadougou’, ‘Africa/Porto-Novo’, ‘Africa/Sao_Tome’, ‘Africa/Timbuktu’, ‘Africa/Tripoli’, ‘Africa/Tunis’, ‘Africa/Windhoek’, ‘America/Adak’, ‘America/Anchorage’, ‘America/Anguilla’, ‘America/Antigua’, ‘America/Araguaina’, ‘America/Argentina/Buenos_Aires’, ‘America/Argentina/Catamarca’, ‘America/Argentina/ComodRivadavia’, ‘America/Argentina/Cordoba’, ‘America/Argentina/Jujuy’, ‘America/Argentina/La_Rioja’, ‘America/Argentina/Mendoza’, ‘America/Argentina/Rio_Gallegos’, ‘America/Argentina/Salta’, ‘America/Argentina/San_Juan’, ‘America/Argentina/San_Luis’, ‘America/Argentina/Tucuman’, ‘America/Argentina/Ushuaia’, ‘America/Aruba’, ‘America/Asuncion’, ‘America/Atikokan’, ‘America/Atka’, ‘America/Bahia’, ‘America/Bahia_Banderas’, ‘America/Barbados’, ‘America/Belem’, ‘America/Belize’, ‘America/Blanc-Sablon’, ‘America/Boa_Vista’, ‘America/Bogota’, ‘America/Boise’, ‘America/Buenos_Aires’, ‘America/Cambridge_Bay’, ‘America/Campo_Grande’, ‘America/Cancun’, ‘America/Caracas’, ‘America/Catamarca’, ‘America/Cayenne’, ‘America/Cayman’, ‘America/Chicago’, ‘America/Chihuahua’, ‘America/Coral_Harbour’, ‘America/Cordoba’, ‘America/Costa_Rica’,

‘America/Creston’, ‘America/Cuiaba’, ‘America/Curacao’, ‘America/Danmarkshavn’, ‘America/Dawson’, ‘America/Dawson_Creek’, ‘America/Denver’, ‘America/Detroit’, ‘America/Dominica’, ‘America/Edmonton’, ‘America/Eirunepe’, ‘America/El_Salvador’, ‘America/Ensenada’, ‘America/Fort_Wayne’, ‘America/Fortaleza’, ‘America/Glace_Bay’, ‘America/Godthab’, ‘America/Goose_Bay’, ‘America/Grand_Turk’, ‘America/Grenada’, ‘America/Guadeloupe’, ‘America/Guatemala’, ‘America/Guayaquil’, ‘America/Guyana’, ‘America/Halifax’, ‘America/Havana’, ‘America/Hermosillo’, ‘America/Indiana/Indianapolis’, ‘America/Indiana/Knox’, ‘America/Indiana/Marengo’, ‘America/Indiana/Petersburg’, ‘America/Indiana/Tell_City’, ‘America/Indiana/Vevay’, ‘America/Indiana/Vincennes’, ‘America/Indiana/Winamac’, ‘America/Indianapolis’, ‘America/Inuvik’, ‘America/Iqaluit’, ‘America/Jamaica’, ‘America/Jujuy’, ‘America/Juneau’, ‘America/Kentucky/Louisville’, ‘America/Kentucky/Monticello’, ‘America/Knox_IN’, ‘America/Kralendijk’, ‘America/La_Paz’, ‘America/Lima’, ‘America/Los_Angeles’, ‘America/Louisville’, ‘America/Lower_Princes’, ‘America/Maceio’, ‘America/Managua’, ‘America/Manaus’, ‘America/Marigot’, ‘America/Martinique’, ‘America/Matamoros’, ‘America/Mazatlan’, ‘America/Mendoza’, ‘America/Menominee’, ‘America/Merida’, ‘America/Metlakatla’, ‘America/Mexico_City’, ‘America/Miquelon’, ‘America/Moncton’, ‘America/Monterrey’, ‘America/Montevideo’, ‘America/Montreal’, ‘America/Montserrat’, ‘America/Nassau’, ‘America/New_York’, ‘America/Nipigon’, ‘America/Nome’, ‘America/Noronha’, ‘America/North_Dakota/Beulah’, ‘America/North_Dakota/Center’, ‘America/North_Dakota/New_Salem’, ‘America/Ojinaga’, ‘America/Panama’, ‘America/Pangnirtung’, ‘America/Paramaribo’, ‘America/Phoenix’, ‘America/Port-au-Prince’, ‘America/Port_of_Spain’, ‘America/Porto_Acre’, ‘America/Porto_Velho’, ‘America/Puerto_Rico’, ‘America/Rainy_River’, ‘America/Rankin_Inlet’, ‘America/Recife’, ‘America/Regina’, ‘America/Resolute’, ‘America/Rio_Branco’, ‘America/Rosario’, ‘America/Santa_Isabel’, ‘America/Santarem’, ‘America/Santiago’, ‘America/Santo_Domingo’, ‘America/Sao_Paulo’, ‘America/Scoresbysund’, ‘America/Shiprock’, ‘America/Sitka’, ‘America/St_Barthelemy’, ‘America/St_Johns’, ‘America/St_Kitts’, ‘America/St_Lucia’, ‘America/St_Thomas’, ‘America/St_Vincent’, ‘America/Swift_Current’, ‘America/Tegucigalpa’, ‘America/Thule’, ‘America/Thunder_Bay’, ‘America/Tijuana’, ‘America/Toronto’, ‘America/Tortola’, ‘America/Vancouver’, ‘America/Virgin’, ‘America/Whitehorse’, ‘America/Winnipeg’, ‘America/Yakutat’, ‘America/Yellowknife’, ‘Antarctica/Casey’, ‘Antarctica/Davis’, ‘Antarctica/DumontDUrville’, ‘Antarctica/Macquarie’, ‘Antarctica/Mawson’, ‘Antarctica/McMurdo’, ‘Antarctica/Palmer’, ‘Antarctica/Rothera’, ‘Antarctica/South_Pole’, ‘Antarctica/Syowa’, ‘Antarctica/Vostok’, ‘Arctic/Longyearbyen’, ‘Asia/Aden’, ‘Asia/Almaty’, ‘Asia/Amman’, ‘Asia/Anadyr’, ‘Asia/Aqtau’, ‘Asia/Aqtobe’, ‘Asia/Ashgabat’, ‘Asia/Ashkhabad’, ‘Asia/Baghdad’, ‘Asia/Bahrain’, ‘Asia/Baku’, ‘Asia/Bangkok’, ‘Asia/Beirut’, ‘Asia/Bishkek’, ‘Asia/Brunei’, ‘Asia/Calcutta’, ‘Asia/Choibalsan’, ‘Asia/Chongqing’, ‘Asia/Chungking’, ‘Asia/Colombo’, ‘Asia/Dacca’, ‘Asia/Damascus’, ‘Asia/Dhaka’, ‘Asia/Dili’, ‘Asia/Dubai’, ‘Asia/Dushanbe’, ‘Asia/Gaza’, ‘Asia/Harbin’, ‘Asia/Hebron’, ‘Asia/Ho_Chi_Minh’, ‘Asia/Hong_Kong’, ‘Asia/Hovd’, ‘Asia/Irkutsk’, ‘Asia/Istanbul’, ‘Asia/Jakarta’, ‘Asia/Jayapura’, ‘Asia/Jerusalem’, ‘Asia/Kabul’, ‘Asia/Kamchatka’, ‘Asia/Karachi’, ‘Asia/Kashgar’, ‘Asia/Kathmandu’, ‘Asia/Katmandu’, ‘Asia/Khandyga’, ‘Asia/Kolkata’, ‘Asia/Krasnoyarsk’, ‘Asia/Kuala_Lumpur’, ‘Asia/Kuching’, ‘Asia/Kuwait’, ‘Asia/Macao’, ‘Asia/Macau’, ‘Asia/Magadan’, ‘Asia/Makassar’, ‘Asia/Manila’, ‘Asia/Muscat’, ‘Asia/Nicosia’, ‘Asia/Novokuznetsk’, ‘Asia/Novosibirsk’, ‘Asia/Omsk’, ‘Asia/Oral’, ‘Asia/Phnom_Penh’, ‘Asia/Pontianak’, ‘Asia/Pyongyang’, ‘Asia/Qatar’, ‘Asia/Qyzylorda’, ‘Asia/Rangoon’, ‘Asia/Riyadh’, ‘Asia/Saigon’, ‘Asia/Sakhalin’, ‘Asia/Samarkand’, ‘Asia/Seoul’, ‘Asia/Shanghai’, ‘Asia/Singapore’, ‘Asia/Taipei’, ‘Asia/Tashkent’, ‘Asia/Tbilisi’, ‘Asia/Tehran’, ‘Asia/Tel_Aviv’, ‘Asia/Thimbu’, ‘Asia/Thimphu’, ‘Asia/Tokyo’, ‘Asia/Ujung_Pandang’, ‘Asia/Ulaanbaatar’, ‘Asia/Ulan_Bator’, ‘Asia/Urumqi’, ‘Asia/Ust-Nera’, ‘Asia/Vientiane’, ‘Asia/Vladivostok’, ‘Asia/Yakutsk’, ‘Asia/Yekaterinburg’, ‘Asia/Yerevan’, ‘Atlantic/Azores’, ‘Atlantic/Bermuda’, ‘Atlantic/Canary’, ‘Atlantic/Cape_Verde’, ‘Atlantic/Faeroe’, ‘Atlantic/Faroe’, ‘Atlantic/Jan_Mayen’, ‘Atlantic/Madeira’, ‘Atlantic/Reykjavik’, ‘Atlantic/South_Georgia’, ‘Atlantic/St_Helena’, ‘Atlantic/Stanley’, ‘Australia/ACT’, ‘Australia/Adelaide’, ‘Australia/Brisbane’, ‘Australia/Broken_Hill’, ‘Australia/Canberra’, ‘Australia/Currie’, ‘Australia/Darwin’, ‘Australia/Eucla’, ‘Australia/Hobart’, ‘Australia/LHI’, ‘Australia/Lindeman’, ‘Australia/Lord_Howe’, ‘Australia/Melbourne’, ‘Australia/NSW’, ‘Australia/North’, ‘Australia/Perth’, ‘Australia/Queensland’, ‘Australia/South’, ‘Australia/Sydney’, ‘Australia/Tasmania’, ‘Australia/Victoria’, ‘Australia/West’, ‘Australia/Yancowinna’, ‘Brazil/Acre’, ‘Brazil/DeNoronha’, ‘Brazil/East’, ‘Brazil/West’, ‘CET’, ‘CST6CDT’, ‘Canada/Atlantic’, ‘Canada/Central’, ‘Canada/East-Saskatchewan’, ‘Canada/Eastern’, ‘Canada/Mountain’, ‘Canada/Newfoundland’, ‘Canada/Pacific’, ‘Canada/Saskatchewan’, ‘Canada/Yukon’, ‘Chile/Continental’, ‘Chile/EasterIsland’, ‘Cuba’, ‘EET’, ‘EST’, ‘EST5EDT’, ‘Egypt’, ‘Eire’, ‘Etc/GMT’, ‘Etc/GMT+0’, ‘Etc/GMT+1’, ‘Etc/GMT+10’, ‘Etc/GMT+11’, ‘Etc/GMT+12’, ‘Etc/GMT+2’, ‘Etc/GMT+3’,

'Etc/GMT+4', 'Etc/GMT+5', 'Etc/GMT+6', 'Etc/GMT+7', 'Etc/GMT+8', 'Etc/GMT+9', 'Etc/GMT-0', 'Etc/GMT-1', 'Etc/GMT-10', 'Etc/GMT-11', 'Etc/GMT-12', 'Etc/GMT-13', 'Etc/GMT-14', 'Etc/GMT-2', 'Etc/GMT-3', 'Etc/GMT-4', 'Etc/GMT-5', 'Etc/GMT-6', 'Etc/GMT-7', 'Etc/GMT-8', 'Etc/GMT-9', 'Etc/GMT0', 'Etc/Greenwich', 'Etc/UCT', 'Etc/UTC', 'Etc/Universal', 'Etc/Zulu', 'Europe/Amsterdam', 'Europe/Andorra', 'Europe/Athens', 'Europe/Belfast', 'Europe/Belgrade', 'Europe/Berlin', 'Europe/Bratislava', 'Europe/Brussels', 'Europe/Bucharest', 'Europe/Budapest', 'Europe/Busingen', 'Europe/Chisinau', 'Europe/Copenhagen', 'Europe/Dublin', 'Europe/Gibraltar', 'Europe/Guernsey', 'Europe/Helsinki', 'Europe/Isle_of_Man', 'Europe/Istanbul', 'Europe/Jersey', 'Europe/Kaliningrad', 'Europe/Kiev', 'Europe/Lisbon', 'Europe/Ljubljana', 'Europe/London', 'Europe/Luxembourg', 'Europe/Madrid', 'Europe/Malta', 'Europe/Mariehamn', 'Europe/Minsk', 'Europe/Monaco', 'Europe/Moscow', 'Europe/Nicosia', 'Europe/Oslo', 'Europe/Paris', 'Europe/Podgorica', 'Europe/Prague', 'Europe/Riga', 'Europe/Rome', 'Europe/Samara', 'Europe/San_Marino', 'Europe/Sarajevo', 'Europe/Simferopol', 'Europe/Skopje', 'Europe/Sofia', 'Europe/Stockholm', 'Europe/Tallinn', 'Europe/Tirane', 'Europe/Tiraspol', 'Europe/Uzhgorod', 'Europe/Vaduz', 'Europe/Vatican', 'Europe/Vienna', 'Europe/Vilnius', 'Europe/Volgograd', 'Europe/Warsaw', 'Europe/Zagreb', 'Europe/Zaporozhye', 'Europe/Zurich', 'GB', 'GB-Eire', 'GMT', 'GMT+0', 'GMT-0', 'GMT0', 'Greenwich', 'HST', 'Hongkong', 'Iceland', 'Indian/Antananarivo', 'Indian/Chagos', 'Indian/Christmas', 'Indian/Cocos', 'Indian/Comoro', 'Indian/Kerguelen', 'Indian/Mahe', 'Indian/Maldives', 'Indian/Mauritius', 'Indian/Mayotte', 'Indian/Reunion', 'Iran', 'Israel', 'Jamaica', 'Japan', 'Kwajalein', 'Libya', 'MET', 'MST', 'MST7MDT', 'Mexico/BajaNorte', 'Mexico/BajaSur', 'Mexico/General', 'NZ', 'NZ-CHAT', 'Navajo', 'PRC', 'PST8PDT', 'Pacific/Apia', 'Pacific/Auckland', 'Pacific/Chatham', 'Pacific/Chuuk', 'Pacific/Easter', 'Pacific/Efate', 'Pacific/Enderbury', 'Pacific/Fakaofo', 'Pacific/Fiji', 'Pacific/Funafuti', 'Pacific/Galapagos', 'Pacific/Gambier', 'Pacific/Guadalcanal', 'Pacific/Guam', 'Pacific/Honolulu', 'Pacific/Johnston', 'Pacific/Kiritimati', 'Pacific/Kosrae', 'Pacific/Kwajalein', 'Pacific/Majuro', 'Pacific/Marquesas', 'Pacific/Midway', 'Pacific/Nauru', 'Pacific/Niue', 'Pacific/Norfolk', 'Pacific/Noumea', 'Pacific/Pago_Pago', 'Pacific/Palau', 'Pacific/Pitcairn', 'Pacific/Pohnpei', 'Pacific/Ponape', 'Pacific/Port_Moresby', 'Pacific/Rarotonga', 'Pacific/Saipan', 'Pacific/Samoa', 'Pacific/Tahiti', 'Pacific/Tarawa', 'Pacific/Tongatapu', 'Pacific/Truk', 'Pacific/Wake', 'Pacific/Wallis', 'Pacific/Yap', 'Poland', 'Portugal', 'ROC', 'ROK', 'Singapore', 'Turkey', 'UCT', 'US/Alaska', 'US/Aleutian', 'US/Arizona', 'US/Central', 'US/East-Indiana', 'US/Eastern', 'US/Hawaii', 'US/Indiana-Starke', 'US/Michigan', 'US/Mountain', 'US/Pacific', 'US/Pacific-New', 'US/Samoa', 'UTC', 'Universal', 'W-SU', 'WET', 'Zulu']

11.3.18. Hierarchy functions

- *fn.hierarchy*: Return the UUID of the current node.
- *fn.hierarchy_parent*: Return the UUID of the parent.
- *fn.hierarchy_path*: Return the current node hierarchy as a list of UUIDs.
- *fn.hierarchy_parent_path*: Return the current node parent hierarchy as a list of UUIDs.
- *fn.hierarchy_friendly_path*: Return the current node hierarchy as a dot-separated hierarchy string.
- *fn.hierarchy_friendly_parent_path*: Return the current node parent hierarchy as a dot-separated hierarchy string.
- *fn.friendly_path_choices*: Return a sorted list of friendly hierarchy paths. Used with a parameter:
  - `down`: all hierarchies paths below, including current hierarchy path
  - `below`: all hierarchies below current hierarchy path, excluding current hierarchy path
  - `local`: current hierarchy path
- *fn.is_site*: Return true or false if run at site context hierarchy or not.
- *fn.authorized_admin_allowed_hierarchy_level*: Given a hierarchy node type parameter, evaluates to `true` or `false` in accordance with the user's authorized admin hierarchy role permissions to the hierarchy node type. Refer to the Authorized Admin Hierarchy Roles topic in the Core Feature Guide.

The hierarchy node type parameter value can be:

- `Customer`
- `Hcs`
- `IntermediateNode`
- `Provider`
- `Reseller`
- `Site`

The `LinkedSite` node type is included in the `Site` node type parameter.

Examples:

Example	Output
{{fn.hierarchy}}	'52162d552afa433946245bcb'
{{fn.hierarchy_parent}}	'52162d522afa433941245ba0'
{{fn.hierarchy_path}}	['1c0efeg2c0deab10da595101', '52162d4c2afa433940245ba3', '52162d4e2afa43393b245ba2', '52162d522afa433941245ba0', '52162d552afa433946245bcb']
{{fn.hierarchy_parent_path}}	['1c0efeg2c0deab10da595101', '52162d4c2afa433940245ba3', '52162d4e2afa43393b245ba2', '52162d522afa433941245ba0']
{{fn.hierarchy_friendly_path}}	'sys.GenCorp.SuperCom.ABCGroup. Branch1'
{{fn.hierarchy_friendly_parent_path}}	'sys.GenCorp.SuperCom.ABCGroup'
Hierarchy = sys.hcs.CS-P.CS-NB.AAAGlobal {# fn.friendly_path_choices ,down #}	["sys.hcs.CS-P.CS-NB.AAAGlobal", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOC001", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOC002", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOC003", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOC004", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOC005", "sys.hcs.CS-P.CS-NB.AAAGlobal. ↪LOCALIZE001"]
Hierarchy = sys.hcs.CS-P.CS-NB.AAAGlobal (cust) {{ fn.is_site }}	false
Can be added as menu or dashboard item condition to hide the item is false. ((fn.authorized_admin_allowed_hierarchy_ ↪level \ Customer == fn.true))	true

11.3.19. User details functions

Function	Description	Example
fn.request_user_name	Return the logged in username.	`{{fn.request_user_name}}`
fn.request_user_role	Return the logged in user role.	`{{fn.request_user_role}}`
fn.request_user_email	Return the logged in user email address.	`{{fn.request_user_email}}`
fn.request_user_pkid	Return the logged in user pkid.	`{{fn.request_user_pkid}}`
fn.request_user_type	Return the logged in user's user type, such as "Admin", "End User" and "End User + Admin".	`{{fn.request_user_type}}`

Function	Description	Example
fn.list_end_user_names	Given a lookup direction and specified target hierarchy type as input parameters, return a list of *end users* (data.User.username) from the current hierarchy scope to the specified level. The user type should contain "End User". The direction can be "up", "down", "above", "below", or "local". Target hierarchy type can be "System", "Hcs", "Provider", "Reseller", "Customer", "IntermediateNode", "Site", "LinkedSite". To exclude target hierarchy type or direction from the resulting filter, pass fn.null.	`{{# fn.list_end_user_names up Customer #}}`
fn.user_type_from_context_det	Given a provided context with user details, typically "input" or "pwf", returns the *user type* for the user based on the role and authorized admin hierarchy association. • User type is "End User + Admin" if role is selfservice and there is an associated authorized admin hierarchy • User type is "End User" when role is selfservice without authorized admin hierarchy • User type is "Admin" when role is administration • Otherwise, user type is Invalid	`{{fn.user_type_from_context_details input }}`

fn.user_vendor_status	Given a username as input parameter, return the vendor(s) of the services that the user has, as follows:
	• `cisco` for Cisco services
	• `microsoft` for Microsoft services
	• `webex` for Webex services
	• `hybrid_cisco_ms` for Cisco and Microsoft services
	• `hybrid_cisco_webex` for Cisco and Webex services
	• `hybrid_ms_webex` for Microsoft and Webex services
	• `None` otherwise
		`{{fn.user_vendor_status abby.black}}`
		Output
		`cisco`

11.3.20. Number management functions

CUSTOMER_INI_ENABLED

The `fn.get_lines` macro function uses the value of a CUSTOMER_INI_ENABLED macro at the relevant hierarchy:

- If `((True))`, then apply the function to the Internal Number Inventory (INI) at the hierarchy: `data.InternalNumberInventory.internal_number`.
- If `((False))`, then apply the function to `device.cucm.Line.pattern`.

The macro function will check the CUSTOMER_INI_ENABLED macro first. This macro should exist at the required hierarchy level and have a value of `((True))` or `((False))`.

Function

Function name: `fn.get_lines`.

Example where `CUSTOMER_INI_ENABLED` macro is `((True))`:

Example	Output
`{{fn.get_lines}}`	`[{"value": "1000", "title": "1000 (Used)  \\+27826543001", {"value": "2000", "title": "2000"}, {"value": "3000", "title": "3000 (Used-Utility)"}, {"value": "4000", "title": "4000 (Cooling) \\+27826543004 ..↪.(release date: 2021-09-25)..."}, {"value": "5000", "title": "5000 (Reserved)"}]`

Parameters

The parameter names and values are listed below.

Note: More than one parameter can be used. These should then be comma-separated, for example: `{{fn.get_lines status:Used,direction:parent}}`.

- `status`: the status of the line.

  Parameter values:

  - `All`
  - `Available`
  - `Used`

  - `Available_or_Used`
  - `Used-Utility`
  - `Used_or_Used-Utility`
  - `Reserved`
  - `Cooling`

  For details, refer to the Number Status and Usage topic in the Core Feature Guide.

  Usage example: `{{fn.get_lines status:Available}}`

Example	Output
`{{fn.get_lines status:Available}}`	`[ {"value": "2000", "title": "2000"} ]`

- `direction`: the line search direction in the hierarchy.

  Note: The default search direction is up, i.e. without the parameter is equal to `direction:up`.

  Parameter values:

  - `up` - Upwards. Include current hierarchy. This is the default if the parameter is not used.
  - `down` - Downwards. Include current hierarchy.
  - `local`: - On this level only. Include current hierarchy.
  - `parent` - Parent only. Exclude current hierarchy, in other words, search the parent as `local`.
  - `below` - Downwards. Exclude current hierarchy.
  - `above` - Upwards. Exclude current hierarchy.

  Usage example: `{{fn.get_lines direction:parent}}`

Example	Output
`{{fn.get_lines direction:parent}}`	`[{"value": "1000", "title": "1000 (Used)  \\+27826543001"}, {"value": "3000", "title": "3000 (Used-Utility)"} ]`

- `partition`: the line search is for `device.cucm.Line.pattern` in the specified partition only, i.e. the specified `routePartitionName`.

  Parameter value: a partition name, for example *Site-REL103-Customer*.

Example	Output
{{fn.get_lines partition:Site-REL103-Customer}}	[{"value": "2000", "title": "2000"}]

- `e164`: the line search is for the E164 number.

  Parameter value: an E164 number, for example `\+27826543001`.

Example	Output
{{fn.get_lines e164:"\+27826543001"}}	[{"value": "1000", "title": "1000 (Used) \\+27826543001"}]

- `scalar_list_only:true`: only display the `value` in the `title, value` pair of the result list.

  Parameter value: only one value: `true`. Without the parameter, the default display is the `title, value` pair.

Example	Output
{{fn.get_lines scalar_list_only:true}}	["1000", "2000", "3000", "4000", "5000"]

Parameter types and permutations

Parameters are of the following types:

- Applies to INI: `status`, `e164`
- Applies to cucm/Line: `partition`
- Other: `scalar_list`, `direction`

The types of parameters will determine the types of lines returned:

- Internal Number Inventory or
- `device.cucm.Line.pattern`

in accordance with whether `CUSTOMER_INI_ENABLED` is `((True))` or `((False))`.

The table below shows the result of parameters and combinations used in these cases.

Note: The default (if not specified) `direction` is up.

Parameter Combination	INI enabled	Result
`partition [scalar_list, direction]`	Y	Return `cucm.Line` in `partition`
`partition [scalar_list, direction]`	N	Return `cucm.Line` in `partition`
`status, partition`	Y	Return INI by `status` upwards
`status, partition`	N	Return `cucm.Line` in `partition`
`status`	Y	Return INI by `status` upwards
`status`	N	Return `cucm.Line` upwards
`partition`	Y	Return `cucm.Line` in `partition`
`partition`	N	Return `cucm.Line` in `partition`

Get DN number from E164 range

fn.get_dn_number: Return the matching Directory Number (DN) in a range given a E164 number as input. The E164 ranges are of 1, 10, 100 and 1000.

Two models are queried. If no results are found in `data/HcsDpDNE164AssociateDAT` then the ranges in `data/HcsDpDNMultiE164AssociateDAT` are queried.

Examples:

Example	Output
hierarchy: sys.hcs.CS.Global.LOC002 `{{ fn.get_dn_number \+121000 }}`	`82041000`

Get a list of site-associated E164 PKIDs

fn.associated_e164_dn_pkids: Given a site hierarchy PKID as input, return the list of E164 PKIDs associated with the site.

Examples:

Example	Output
Site hierarchy PKID: 61a67eb627e8f5534d6366a1 `{{ fn.associated_e164_dn_pkids 61a67eb627e8f5534d6366a }}`	`['61a6c5df008014baaa296762', '61a6c5e0008014baaa296777', '61a6c5e0008014baaa29678c']`

Get a list of associated DNs

fn.associated_dn_list: Given an optional hierarchy as input, return the list of DNs associated with an E164 number at the hierarchy.

Without the hierarchy PKID, all DNs associated with an E164 number are listed.

Examples:

Example	Output
Hierarchy PKID: 61edc1c122a4e9dde5664cee `{# fn.associated_dn_list 61edc1c122a4e9dde5664cee #}`	`["85011008", '85011009", '85022008"]`

11.3.21. Phone functions

Function names:

- *fn.get_phone_status*

  Given as input parameters:

  - a phone PKID
  - followed by a comma and then exactly one RIS API field name.

    The fields below are for example used in the VOSS Automate Admin Portal list view of Phones:

    * `status`
    * `ip_address`
    * `cm_node`

    To see a full list of available fields, refer to the Cisco RIS API documentation.

  Returns:

  A string with the results according to the selected field name

  Examples:

Example	Output
{{fn.get_phone_status 5ca2b90bce894e0014d488fb, Status}}	"Registered"

- *fn.get_phone_statuses*

  Given as input parameters:

  – a semicolon separated list of phone PKIDs

  – optionally followed by a comma and then a semicolon separated list of RIS API field names.

    The fields below are for example used in the VOSS Automate Admin Portal list view of Phones:

    * status
    * ip_address
    * cm_node

    To see a full list of available fields, run the macro function without RIS API field names or refer to the Cisco RIS API documentation.

  Returns:

  A list of phone status results with details:

  – containing either all fields if no optional field name list was provided, or

  – if an optional field name list was provided, results according to the selected field names.

Note: This function will query the Unified CM for the requested PKIDs and return the phone registration status of these.

Examples:

Example	Output

{{fn.get_phone_statuses
5ca2b90bce894e0014d488fb;
3da7c60bd4632f1113a255dc,
Status;
ip_address}}
	[{
	"Status": "Registered",
	"ip_address": "172.29.90.80"
	},
	{
	"Status": "Registered",
	"ip_address": "172.29.90.11"
	}]

- *fn.get_hierarchy_phone_statuses*

  A function to retrieve phone statuses as JSON objects with the phone name as the key. including those not returned by the Call Manager API

Given as input parameters:

- `status` - a phone status:

  The `status` parameter value is case insensitive, for example `Unregistered` will also find `UnRegistered`.

  * Registered
  * Unregistered
  * PartiallyRegistered
  * Rejected
  * Any
  * Unknown
  * None - will return only phones not returned by the Unified CM API; the only data will be hierarchy, phone name and status (None).

- optionally followed by value of `cucm` (host or pkid)

  Note:

  * If the function is run at a hierarchy *above* customer level, the parameter is mandatory.
  * The function will yield an error if more than one `cucm` host has the same name. Use the `pkid` as value in this case.

- optionally followed by value of `device_name` (phone name prefix: "startswith" match, e.g. SEP, BOT. . .)
- optionally followed by value of `ip_address` (IP address: partial "startswith" match, e.g. "10.10." is valid)
- optionally followed by value of `dir_number` (directory number: "startswith" match, does not have to be the whole number)

  semicolon separated list of RIS API field names.

  The fields below are for example used in the VOSS Automate Admin Portal list view of Phones:

  * `status`
  * `ip_address`
  * `cm_node`

  To see a full list of available fields, run the macro function without RIS API field names or refer to the Cisco RIS API documentation.

Returns:

A list of phone status results with details:

- containing either all fields if no optional field name list was provided, or
- if an optional field name list was provided, results according to the selected field names.

Examples:

Example	Output (truncated)
{{fn.get_hierarchy_phone_statuses Registered}}	{ "CTICTIREM": { "hierarchy": "sys.hcs.CS-P.CS-NB.AAAGlobal", "cm_node": "cucm", "ip_address": "172.29.90.80", "status": "Registered with cucm", "Name": "CTICTIREM", "DirNumber": "82025608-Registered", "DeviceClass": "Phone", [...]

- *fn.save_hierarchy_phone_statuses*

  A function to retrieve phone statuses as a Comma Separated Value (CSV) file that is stored in the database as an instance of `data/File` and accessible from the **File Management** menu on the GUI. The CSV file can for example be exported (as JSON with `.zip` extension, containing JSON and CSV files) and managed in a spreadsheet.

  The `status` parameter value is case insensitive, for example `Unregistered` will also find `UnRegistered`.

  The CSV filename format *with* a parameter value for `<fname_prefix>` is:

  – *<fname_prefix>_<hierarchy_name>_<status>_<timestamp>_<number of entries>.csv*

  otherwise, the CSV filename format is:

  – *<hierarchy_name>_<status>_<timestamp>_<number of entries>.csv*

  The *<timestamp>* format is: `%Y%m%d_%H%M%S`

  Input parameters as with *fn.get_hierarchy_phone_statuses*, and:

  – optionally followed by a value of `fname_prefix` (CSV filename prefix)

  Examples:

Example	Output (also refer to data/File instance)
{{fn.save_hierarchy_phone_statuses Registered}}	{ "url": "/api/data/File/5bf43f83affa931c2505f6a6/", "description": "AAAGlobal_Registered_20201014)161506_3" }

11.3.22. Localization functions

- *fn.localize*: Return a value that is localized, in other words it will be translated if a translation exists. It is used with a macro call that returns a value from a data store.
- *fn.list_installed_languages*: List the installed languages on the system. This includes languages in the Admin Portal and selfservice GUI.
- *fn.list_installed_languages_admin*: List the installed languages in the Admin Portal.
- *fn.list_installed_languages_selfservice*: List the installed languages in the selfservice GUI.
- *fn.list_installed_languages_by_role*: List the installed languages based on the User Role Interface value.

  The User Role Interface value is a parameter of this function:

  - admin - installed languages in the Admin Portal
  - self-service - selfservice GUI
  - none - union of admin and selfservice languages
- *fn.localize_choices*: Given a parameter containing a list of strings, return title-value pairs of the strings, with the titles marked for localization. The function is used to localize drop-down lists. For example, given a list:

```
[ 'choice1', 'choice2' ]
```

  then the function will return

```
[ {'title': _('choice1'), 'value': 'choice1'},
  {'title': _('choice2'), 'value': 'choice2'}]
```

  which is then localized upon rendering. For English, this will simply be:

```
[ {'title': 'choice1', 'value': 'choice1'},
  {'title': 'choice2', 'value': 'choice2'}]
```

Example	Output
`{{ fn.localize data.LocalizedModelStore.localized_value | where_clause }}`	A localized value is returned.
`{# fn.list_installed_languages #}`	`[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title':'German'}]`
`{# fn.list_installed_languages_admin #}`	`[{'value':'en-us', 'title': 'English'}]`
`{# fn.list_installed_languages_selfservice #}`	`[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title': 'German'}]`
`{# fn.list_installed_languages_by_role none #}`	`[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title': 'German'}]`
`{{ fn.localize_choices data.HcsNbnSupportedModelsDAT.operations | modelType:data/User }}`	output before localization: `[{'title': _('create'), 'value': 'create'}, {'title': _('delete'), 'value': 'delete'}, {'title': _('update'), 'value': 'update'}]`

11.3.23. Log functions

- *fn.log*: Given a log level and message, display it in the log. Log levels can be: debug, critical, warn, error or info.
- *fn.txn_log*: Given a message, display it in the transaction log on the Admin Portal.

  The macro is typically added to a workflow “Set” step for debugging purposes.

Examples:

Example	Output

`{{fn.log info, This is an informational message.}}`
 	
`{{fn.log debug, Debug message.}}`
	In app.log:
	`INFO This is an informational message.`
	
	`DEBUG Debug message.`

A workflow (1PWF) step Set variable "bar" is a message with the value of "name". `{{fn.txn_log pwf.name}}`	Step 1 - Set workflow context (1PWF): `{ "set_list": [ { "set_var_name": "name", "set_var_value": "foo" }, { "set_var_name": "bar", "set_var_value": "{{fn.txn_log pwf.name}}" } ], "bar": "foo", "name": "foo" }`

For examples of the macros below, refer to the example for Custom Messages and Details in Provisioning Workflows.

- `fn.set_current_transaction_detail(<detail_text>)`

  To customize the transaction detail of the *current* transaction: top (ancestor) or child transaction. If a transaction fails, this detail will also override the standard detail.

- `fn.set_top_level_transaction_detail(<detail_text>)`

  To customize the transaction detail of the top (ancestor) transaction. If there is no top transaction, it will customize the current transaction detail. If a transaction fails, this detail will also override the standard detail.

- `fn.set_current_transaction_message(<message_text>)`

  To customize the transaction message of the *current* transaction: top or child transaction. If a transaction fails, this message will *not* override the standard error message.

- `fn.set_top_level_transaction_message(<message_text>)`

  To customize the transaction message of the top transaction. If there is no top transaction, it will customize the current transaction message. If a transaction fails, this message will *not* override the

standard error message.

11.3.24. Object functions

- *fn.is_object*: Return true or false if the parameter is an object or not.
- *fn.object_keys*: Given an object and additional optional parameter, return the list of keys that match the parameter value, or all the keys if no parameter value is given.

Example object:

```
{
  "input": {
     "object": {
       "boolean_1": true,
       "boolean_2": false,
       "boolean_3": true,
       "string_1": "1",
       "string_1_dup": "1",
       "string_2": "2",
       "integer_1": 1,
       "integer_1_dup": 1,
       "integer_2": 2

  }
}
```

Examples

Example	Output
{{ fn.object_keys input.object,true }}	["boolean_1","boolean_3"]
{{fn.object_keys input.object,"1"}}	["string_1","string_1_dup"]
{{fn.object_keys input.object,1}}	["integer_1","integer_1_dup"]
{{fn.object_keys input.object}}	["boolean_1","boolean_2", "boolean_3","string_1", "string_1_dup","string_2", "integer_1","integer_1_dup", "integer_2"]

- *fn.object_empty*: Returns and empty object

Example

Example	Output
`{{ fn.object_empty }}`	`{}`

- *fn.object_update* - Given an existing object and a key-value pair, updates and returns the given object with the key and value.
  - If the key dows not exist, the pair is added.
  - If the key exists, the value is updated.

Example

Example	Output

my_object:
`{ "existing_key": "some_value" }`
function call:
`{{ fn.object_update "key","1234", input.my_object }}`
	`{ "existing_key": "some_value", "key": "1234" }`

my_object:
`{ "key": "some_value" }`
function call:
`{{ fn.object_update "key","1234", input.my_object }}`
	`{ "key": "1234" }`

- *fn.object_compare_specific*:

  This macro can for example be used to check previous and input context differences, providing a way to skip long updates on models if the comparison is true.

  Note: If we have fields that do not exist on the left or right side, these will be ignored/skipped and the result will be true.

  Given:

  - a list of fields we need to test for
  - a left object (dictionary) to compare against the right object

- a right object (dictionary) that is compared with left object
- a boolean flag: “true” to ignore nulls or blanks in the objects The flag defaults to False if any value other than ‘true’ is entered

Return:

A dictionary with the result of the comparison. If the comparison is false, return the Field the comparison failed on.

Examples:

Given the same example input below, the output of the two examples below is respectively `"result": False` or `"result":True` depending on the final input parameter: whether empty and blank fields should not or should be ignored.

input:

```
context = {
    "pwf": {
        "left_object": {
            "City": "Bellville",
            "FirstName": "Paul",
            "LastName": "Smith",
            "Department": None,
            "Age": ""
        },
        "right_object": {
            "City": "Bellville",
            "FirstName": "Paul",
            "LastName": "Smith",
            "Department": None,
            "Age": "20"
        },
        "field_list": [
            "City",
            "FirstName",
            "LastName",
            "DoesNotExist",
            "Age",
            "Department"
        ]
    }
}



1. command (don't ignore nulls or blanks):

  ::

     {{ fn.object_compare_specific pwf.field_list, pwf.left_object, pwf.right_object,
↪ false }}

  output:
```

(continues on next page)

(continued from previous page)

::


```
{
"right_object": {
    "Department": None,
    "City": "Bellville",
    "Age": "20",
    "FirstName": "Paul",
    "LastName": "Kruger"
},
"field_list_to_compare": [
    "City",
    "FirstName",
    "LastName",
    "DoesNotExist",
    "Age",
    "Department"
],
"ignore_null_and_blank": False,
"left_object": {
    "Department": None,
    "City": "Bellville",
    "Age": "",
    "FirstName": "Paul",
    "LastName": "Kruger"
},
"result": False,
"field_match_results": [
    {
        "field": "City",
        "left_value": "Bellville",
        "match": True,
        "right_value": "Bellville"
    },
    {
        "field": "FirstName",
        "left_value": "Paul",
        "match": True,
        "right_value": "Paul"
    },
    {
        "field": "LastName",
        "left_value": "Kruger",
        "match": True,
        "right_value": "Kruger"
    },
    {
        "field": "DoesNotExist",
        "match": "Field not found, skip"
    },
    {
```


(continues on next page)

(continued from previous page)

```
            "field": "Age",
            "left_value": "",
            "match": False,
            "right_value": "20"
        },
        {
            "field": "Department",
            "left_value": None,
            "match": True,
            "right_value": None
        }
    ]
    }
```

2. command (ignore nulls or blanks):

::

```
    {{ fn.object_compare_specific pwf.field_list, pwf.left_object, pwf.right_object,
↪ true }}
```

output:

::

```
    {
    "right_object": {
        "Department": None,
        "City": "Bellville",
        "Age": "20",
        "FirstName": "Paul",
        "LastName": "Kruger"
    },
    "field_list_to_compare": [
        "City",
        "FirstName",
        "LastName",
        "DoesNotExist",
        "Age",
        "Department"
    ],
    "ignore_null_and_blank": True,
    "left_object": {
        "Department": None,
        "City": "Bellville",
        "Age": "",
        "FirstName": "Paul",
        "LastName": "Kruger"
    },
    "result": True,
```

(continues on next page)

(continued from previous page)


```
    "field_match_results": [
        {
            "field": "City",
            "left_value": "Bellville",
            "match": True,
            "right_value": "Bellville"
        },
        {
            "field": "FirstName",
            "left_value": "Paul",
            "match": True,
            "right_value": "Paul"
        },
        {
            "field": "LastName",
            "left_value": "Kruger",
            "match": True,
            "right_value": "Kruger"
        },
        {
            "field": "DoesNotExist",
            "match": "Field not found, skip"
        },
        {
            "field": "Age",
            "match": "Field null or blank, skip"
        },
        {
            "field": "Department",
            "match": "Field null or blank, skip"
        }
    ]
    }
```


11.3.25. Conversion functions

- *fn.pkid_to_bkey*: Given a pkid, return the business key.
- *fn.bkey_to_pkid*: Given a business key, return the pkid Provide the data type as an argument.
- *fn.from_business_key_format*: Convert a field business key string format to a list
- *fn.as_int*: Given a string, return an integer.
- *fn.as_string*: Given an integer, return a string.
- *fn.as_bool*: Given strings “True”, “TRUE”, “T”, “t”, “1”, return True. Given strings “False”, “FALSE”, “F”, “f”, “0”, return False.
- *fn.as_list*: Given input, return it as a list. List input is returned as is.
- *fn.int_to_hex*: Return the hexadecimal value of an integer. Application example: for gateways that use hexadecimal values for port numbers.

- *fn.hex_to_int*: Return the integer value of a hexadecimal value. Application example: for gateways that use hexadecimal values for port numbers.
- *fn.getMajorMinorVersion*: Given a version in various formats, return a version of the format <major>.<minor>.
- *fn.list_to_string*: Given a list as input, return the list as a string.

  Note the following in the output - refer to the example below:

  – strings in the list will be prefixed with a Unicode character `u`

  – the "`[`" and "`]`" as well as the commas between list items are included in the converted string
- *fn.list_items_to_string*: Given a list as input (strings or integers), return each item as a string.

  Only simple lists are supported.
- *fn.list_items_to_int*: Given a list of integers as input, return each item as a string.

  Only simple lists are supported.

  Only lists of integers are supported.

Examples:

Example	Output

`macro: USA_pkid =`
` {{data.Countries.__pkid`
`  iso_country_code:USA}}`
 	
`{{fn.pkid_to_bkey macro.USA_pkid}}`
	`"[u'United States of America',`
	`  u'USA', u'']"`

`macro: a_country_bkey =`
` {{data.Countries.__bkey`
`  __pkid:macro.USA_pkid}}`
 	
`{{fn.bkey_to_pkid  macro.a_country_bkey,`
`  data/Countries}}`
	`"52d3eba8893d57373f842acb"`

context data:
`{`
`  "self": {`
`  "bk": "[\"10.110.21.101\",`
`  \"8443\",`
`  \"hcs.CS-P.CS-NB.AAAGlobal\"]"`
`  }`
`}`
function:
`{{ fn.from_business_key_format self.bk }}`
	`[`
	`  "10.110.21.101",`
	`  "8443",`
	`  "hcs.CS-P.CS-NB.AAAGlobal"`
	
	`[`

Example	Output
{{fn.as_int "1"}}	1
{{fn.as_string 12345}}	"12345"
{{fn.as_bool "T"}} {{fn.as_bool "0"}}	true false
{#fn.as_list foo bar#} {#fn.as_list 'foo,bar'#} {#fn.as_list ['foo','bar']#}	['foo bar'] ['foo,bar'] ['foo','bar']
{{fn.int_to_hex 255}}	ff
{{fn.hex_to_int ff}}	255

Example	Output
{{ fn.getMajorMinorVersion 10.5(4) }} {{ fn.getMajorMinorVersion 11.5(1) SU1 }} {{ fn.getMajorMinorVersion 11.5.3 }}	10.5 11.5 11.5

{"pwf": {
"my_list": [1, null, 2,"test", 3, null]
}
}
 	
{{ fn.list_to_string pwf.my_list }}
	"[1, None, 2, u'test', 3, None]"

{"pwf": {
"my_list": [1, 2, "3", 4]
}
}
 	
{{ fn.list_items_to_string pwf.my_list }}
	['1', '2', '3', '4']

{"pwf": {
"my_list": [1, 2, "3", 4]
}
}
 	
{{ fn.list_items_to_int pwf.my_list }}
	[1, 2, 3, 4]

11.3.26. HTTP functions

Note: The macro functions: `fn.request_get` and `fn.perform_http_get` are only available to API users and administrators that have an Access Profile set to **Full Access**.

- *fn.request_get*: Return in JSON format the response of an HTTP request. The HTTP request must start with `http://localhost`

Example request:

```
{{ fn.request_get  http://localhost/api/data/Countries/properties }}
```

The output can be assigned to a variable so that properties can be referenced.

Example with output snippet:

```
http://localhost/api/data/Countries/properties
```


```
{
    "meta": {
        "query": "/api/data/Countries/properties/?hierarchy=[hierarchy]&format=json"
    },
    "choices": [
        [
            "cli_on_prefix",
            "cli_on_prefix"
        ],
        [
            "country_name",
            "country_name"
        ],
        [
...
```


Example of instance output with GET request for an instance with `[pkid]`:

```
http://localhost/api/data/Countries/54e1de60edec65160652e402
```


```
...

  ],
    "business_key": {
        "hierarchy": true,
        "unique": [
            "country_name",
            "iso_country_code"
        ]
    },
    "tagged_versions": []
},
"data": {
    "iso_country_code": "MEX",
     "pstn_access_prefix": "9",
     "pkid": "54e1de60edec65160652e403",
     "default_user_locale": "English United States",
     "network_locale": "United States",
     "standard_access_prefix": "0",
     "international_access_prefix": "00",
     "country_name": "Mexico",
     "international_dial_code": "52",
     "emergency_access_prefix": "066",
     "national_trunk_prefix": "01"
    }
```


- *fn.perform_http_get*: Given a URL parameter and optionally username and password parameters,

return in HTTP status code of the HTTP request.

Example request:

```
{{ fn.perform_http_get  http://<hostname> }}
```

Example request with additional parameters:

```
{{ fn.perform_http_get  http://<hostname> <username> <password>}}
```

Example output:

```
200
```

If the HTTP request raises an error (for example: not found, connection, timeout, and so on), this error will be returned.

11.3.27. Email functions

- *fn.email_html*: Given the parameters:
  - `to`: addressee email address
  - `from`: sender email address
  - `template`: email template

  send an email if an SMTP server is set up at the hierarchy from which the function is called.

  The function is typically used in provisioning workflows to send an email to a subscriber using the Quick Add Subscriber feature if this has been set up - see: *Global settings*.

  Syntax:

  `{{ fn.email_html pwf.to,pwf.from,pwf.template }}`

  where:

  - the parameters have been assigned values in the workflow.

  Workflow snippet example:

```
{
   "entity": "data/ProvisioningWorkflow",
   "entity_type": "model",
   "method": "set",
   "set_list": [
     {
         "set_var_name": "to",
         "set_var_value": "{{ pwf.email_address }}"
     },
     {
         "set_var_name": "email_template",
         "set_var_value": "Quick Add Subscriber"
     },
     {
         "set_var_name": "from",
```

(continues on next page)

(continued from previous page)

```
        "set_var_value": "{{ data.EmailHtmlTemplate.from | name:pwf.email_template␣
↪| direction:up, limit:1 }}"
    },
    {
        "set_var_name": "sendmail",
        "set_var_value": "{{ fn.email_html pwf.to,pwf.from,pwf.email_template }}"
    }
  ],
    "advanced_find_search_direction": "full_tree"
}
```

11.4. Macro Examples and Use

11.4.1. Macro examples - simple

Simple macros must always resolve to one value only.

```
{{data.Countries.iso_country_code | country_name:'South Africa'}}

'ZAF'
```

Call to a non-existent macro.

```
{{macro.DoesNotExist}}

{u'code': 6003,
 u'http_code': 400,
 u'message': u'Macro lookup of macro.DoesNotExist failed at hierarchy sys',
 u'transaction': None}
```

First Partition member name in CSS PSTN-CSS-Cape-Town.

```
{{ device.cucm.Css.members.member.0.routePartitionName | name: 'PSTN-CSS-Cape-Town' }}

'PHONES-PT-Cape-Town'
```

First Partition member UUID in CSS PSTN-CSS-Cape-Town.

```
{{ device.cucm.Css.members.member.0.uuid | name: 'PSTN-CSS-Cape-Town' }}

'{7AF255DC-3A05-A1B4-9E5E-95CD48C3C95F}'
```

11.4.2. Macro examples - list macro

Syntax for List macros is between {# #}. The results are in a list format: comma separated results and between [ ] *All* fields in Countries model.

```
{# data.Countries.* #}

[{u'cli_on_prefix': u'',
  u'country_name': u'Australia',
  u'data_type_': u'data/Countries',
  u'default_user_locale': u'English United States',
  u'emergency_access_prefix': u'000',
  u'international_access_prefix': u'011',
  u'international_dial_code': u'61',
  u'iso_country_code': u'AUS',
  u'national_trunk_prefix': u'0',
  u'network_locale': u'United States',
  u'premium_access_prefix': u'8',
  u'pstn_access_prefix': u'9',
  u'service_access_prefix': u'13'},
 {u'cli_on_prefix': u'',
  u'country_name': u'Bahrain',
  u'data_type_': u'data/Countries',
  u'default_user_locale': u'English United States',
  u'emergency_access_prefix': u'999',
  u'international_access_prefix': u'00',
  u'international_dial_code': u'973',
  u'iso_country_code': u'BHR',
  u'national_trunk_prefix': u'',
  u'network_locale': u'United States',
  u'premium_access_prefix': u'',
  u'pstn_access_prefix': u'9',
  u'service_access_prefix': u''},
.........]
```

Selected fields in Countries model.

```
{# data.Countries.country_name, iso_country_code #}

[{u'country_name': u'Australia',
  u'iso_country_code': u'AUS'},
 {u'country_name': u'Bahrain',
  u'iso_country_code': u'BHR'},
 {u'country_name': u'Canada',
  u'iso_country_code': u'CAN'},
 {u'country_name': u'Denmark',
  u'iso_country_code': u'DNK'},
 ...
 ...
 {u'country_name': u'United States of America',
  u'iso_country_code': u'USA'}
]
```

Specifying one field in the list will return only a list of values and not a key-value pair list.

```
{# data.Countries.country_name #}

[u'Australia',
 u'Bahrain',
 u'Canada',
 ...
 ...
 u'United States of America']
```

Device types: a list of all line patterns in the null partition.

```
{# device.cucm.Line.pattern,routePartitionName | routePartitionName:'NullPartition'#}

[{u'pattern': u'55554444',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'8100240105',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'5544332211',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'55667722',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'8765653',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'66776767',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'3009',
  u'routePartitionName': u'NullPartition'},
 {u'pattern': u'656574747',
  u'routePartitionName': u'NullPartition'},
 ...
 ...  ]
```

Nested structures.

```
{# device.cucm.Css.* | name: 'PSTN-CSS-Cape-Town'#}

[{u'clause': u'PHONES-PT-Cape-Town:PSTN-PT-Cape-Town:
               Pickup-PT-Cape-Town:CallPark-PT-Cape-Town',
  u'hierarchy': u'5171010ecc2e19483c11291b',
  u'members': {u'member': [{u'index': 1,
                            u'routePartitionName': u'PHONES-PT-Cape-Town',
                            u'uuid': u'{7AF255DC-3A05-A1B4-9E5E-95CD48C3C95F}'},
                           {u'index': 2,
                            u'routePartitionName': u'PSTN-PT-Cape-Town',
                            u'uuid': u'{5FA76732-0074-108A-3A91-23D7C6CAC2E1}'},
                           {u'index': 3,
                            u'routePartitionName': u'Pickup-PT-Cape-Town',
                            u'uuid': u'{F789964F-C95D-4095-F6C7-48E587CBFAD8}'},
                           {u'index': 4,
                            u'routePartitionName': u'CallPark-PT-Cape-Town',
                            u'uuid': u'{B4817113-0F32-6E7F-67B2-20645CFC4509}'}]},
```

(continues on next page)

(continued from previous page)

```
   u'name': u'PSTN-CSS-Cape-Town',
   u'partitionUsage': u'General',
   u'uuid': u'{E678A23E-866A-7CE8-AD0F-8AF138E10A18}'}]
```

```
{# device.cucm.Css.name,members | name: 'PSTN-CSS-Cape-Town'#}

[{u'members': {u'member': [{u'index': 1,
                            u'routePartitionName': u'PHONES-PT-Cape-Town',
                            u'uuid': u'{7AF255DC-3A05-A1B4-9E5E-95CD48C3C95F}'},
                           {u'index': 2,
                            u'routePartitionName': u'PSTN-PT-Cape-Town',
                            u'uuid': u'{5FA76732-0074-108A-3A91-23D7C6CAC2E1}'},
                           {u'index': 3,
                            u'routePartitionName': u'Pickup-PT-Cape-Town',
                            u'uuid': u'{F789964F-C95D-4095-F6C7-48E587CBFAD8}'},
                           {u'index': 4,
                            u'routePartitionName': u'CallPark-PT-Cape-Town',
                            u'uuid': u'{B4817113-0F32-6E7F-67B2-20645CFC4509}'}]},
  u'name': u'PSTN-CSS-Cape-Town'}]
```

```
{# device.cucm.Css.name,members.member | name: 'PSTN-CSS-Cape-Town'#}

[{u'members.member': [{u'index': 1,
                       u'routePartitionName': u'PHONES-PT-Cape-Town',
                       u'uuid': u'{7AF255DC-3A05-A1B4-9E5E-95CD48C3C95F}'},
                      {u'index': 2,
                       u'routePartitionName': u'PSTN-PT-Cape-Town',
                       u'uuid': u'{5FA76732-0074-108A-3A91-23D7C6CAC2E1}'},
                      {u'index': 3,
                       u'routePartitionName': u'Pickup-PT-Cape-Town',
                       u'uuid': u'{F789964F-C95D-4095-F6C7-48E587CBFAD8}'},
                      {u'index': 4,
                       u'routePartitionName': u'CallPark-PT-Cape-Town',
                       u'uuid': u'{B4817113-0F32-6E7F-67B2-20645CFC4509}'}],
  u'name': u'PSTN-CSS-Cape-Town'}]
```

```
{# device.cucm.Css.name,members.member.2 | name: 'PSTN-CSS-Cape-Town'#}

[{u'members.member.2': {u'index': 3,
                        u'routePartitionName': u'Pickup-PT-Cape-Town',
                        u'uuid': u'{F789964F-C95D-4095-F6C7-48E587CBFAD8}'},
  u'name': u'PSTN-CSS-Cape-Town'}]
```

```
{# device.cucm.Css.name,members.member.2.routePartitionName | name:
  'PSTN-CSS-Cape-Town'#}

[{u'members.member.2.routePartitionName': u'Pickup-PT-Cape-Town',
  u'name': u'PSTN-CSS-Cape-Town'}]
```

11.4.3. Macro- and macro function nesting

Macros and macro functions can be used as arguments of macros and macro functions. Consider the examples below:

1. Define a macro Masklen as {{fn.length This is a valid string}}.
2. Define a macro as {{fn.mask X macro.Masklen 0}}.
3. The result is evaluated as 'XXX...' to the length of 'This is a valid string'.

11.4.4. Conditional logic macro function

Conditional logic in macros is supported by the `fn.conditional_logic` function that takes two parameters:

- the name of an instance of the `data/ConditionalLogic` data model
- a value that serves as input to `data/ConditionalLogic` the data model.

The namespaces that can be used as `left_expression` values in the data model can depend on the reference to the data model in a Provisioning Workflow. The namespaces - that can also be referenced in full - include:

- `{{self}}`
- `{{previous}}`
- `{{input}}`
- `{{cft}}`
- `{{pwf}}`

Consider the following example `data/ConditionalLogic` data model called "Is_SLC_Allowed":

```
"conditions": [
    {
        "unary_operator": "NOT",
        "right_expression": "{{ logic.DATA }}",
        "conditional_operator": "AND",
        "condition": "contains",
        "left_expression": "{{ pwf.SLCS }}"
    },
    {
        "unary_operator": "NOT",
        "right_expression": "{{ input.CURRENT_SLC }}",
        "conditional_operator": "AND",
        "condition": "containsStartOf",
        "left_expression": "{{ logic.DATA }}"
    },
    {
        "right_expression": "{{ input.CURRENT_SLC }}",
        "condition": "containsStartsWith",
        "unary_operator": "NOT",
        "left_expression": "{{ logic.DATA }}"
    }
]
```

Also suppose the Provisioning Workflow for this example has a list variable `SLCS` and receives an `input.CURRENT_SLC` value.

Furthermore, during the call of the `fn.conditional_logic` function in the Provisioning Workflow, it receives a *scalar* value as an argument, for example:

```
{{ fn.conditional_logic Is_SLC_Allowed,128 }
```

- The scalar value reference `{{ logic.DATA }}` can be ommitted from either `left_expression` or `right_expression`. Its reference is then assumed.
- The input value is referenced as `{{ input.CURRENT_SLC }}`
- The list that is the Provisioning Workflow variable, is `{{ pwf.SLCS }}`

As another example, consider a `data/ConditionalLogic` model called "TestData" with three conditions:

```
"conditions": [
                {
                    "conditional_operator": "OR",
                    "left_expression": "{{input.DATA}}",
                    "condition": "contains",
                    "right_expression": "AAA"
                },
                {
                    "conditional_operator": "AND",
                    "left_expression": "{{input.DATA}}",
                    "condition": "contains",
                    "right_expression": "BBB"
                },
                {
                    "left_expression": "{{input.DATA}}",
                    "condition": "contains",
                    "right_expression": "CCC",
                    "unary_operator": "NOT"
                }
```

The following function checks if a received input value "AAAaaaBBBaaaCCc" fulfills the condition: contains "AAA" OR "BBB" AND NOT "CCC", as in the macro test using a scalar value:

```
{{ fn.conditional_logic TestData,AAaaaBBBaaaCCc }}
```

The condition resolves to true.

Finally in the following example, the conditional function is used as a condition in a Provisioning Workflow. The Data Model instance of data/ConditionalLogic called "Does Newland Exist" tests a single string matching condition:

```
"data": {
    "conditions": [
        {
            "right_expression": "Newland",
            "condition": "isexactly",
            "left_expression": "{{pwf.EXIST}}"
        }
    ],
    "name": "Does Newland Exist"
}
```

The Provisioning Workflow step to apply a Configuration Template if the condition is false. So the step is carried out only if there is not already a `country_name` called “Newland”.

```
"workflow": [
  {
    "templates": [
      {
        "conditions": [
          {
            "condition": "(( fn.conditional_logic \"Does Newland Exist\" == False ))"
          }
        ],
        "template": "CFT1"
      }
    ],
    "entity": "data/Countries",
    "set_list": [
      {
        "set_var_name": "EXIST",
        "set_var_value": "{{data.Countries.country_name|country_name:Newland}}"
      }
    ],
    "method": "add",
    "entity_type": "model"
  }
```

11.4.5. Add a macro inline

To create a macro inline, enter the macro directly in:

1. Default value in a Data Model.
2. Default value in a Configuration Template.
3. Condition of an Operation in a Provisioning Workflow.

11.4.6. Add a macro for re-use

To create a macro for re-use:

1. Choose the hierarchy to which the macro will apply.
2. Choose **data**/**Macro**.
3. Click **Add**.
4. Enter a Name for the macro.
5. Enter the macro in the Macro input box.
6. Click **Save** to save the macro.
7. The Macro is available for editing from data/Macro/{macro name}, and for use in for example a Configuration Template with a macro reference of the form {{macro.macro name}}.

To call a macro created for re-use:

Macros that have been created and saved can be called using macro syntax but with the namespace “macro” followed by the macro name: “{{macro.macroname}}”.

11.4.7. Create a value substitution macro

To write a value substitution macro:

1. Determine where to enter the macro: in-line or for re-use.
2. Determine the reference of the value to resolve:
3. Data Model reference syntax is “data/datamodel.attribute” or “datamodel.attribute”. The first instance of the datamodel.attribute will be the target.
4. Enter any static text that should combine with the evaluated macro - if required. Static text is entered outside the “{{” and “}}”.

11.4.8. Substitution macro examples

```
{{CallManager.host}}

http://{{CallManager.host}}

http://{{CallManager.host}}/{{CallManager.username}}
```

11.4.9. Add an evaluation macro

To write an evaluation macro:

1. Identify tests and result values:
2. A simple test resolves to True or False.
3. an If-Then-Else test resolves to a value.
4. For a simple test, identify the values and operator to resolve to True or False.
5. For an If-Then-Else test, identify If-, Else- and default conditions and values.

11.4.10. Evaluation macro examples

```
((DATA1.d1 == y))
```

Explanation: this macro evaluates to true or false if DATA1.d1 is equal to y.

```
((EVALUATE1.val == y)) <{{CallManager.host}}-Enabled>
 ((EVALUATE1.val == n)) <{{CallManager.host}}-Disabled>
 <{{CallManager.host}}-Not set>
```

Explanation: this macro evaluates data model called “EVALUATE1” with attribute “val” - to value of data model called “CallManager” and with attribute “host”:

- Appended with “-Enabled” if EVALUATE1.val is y
- Appended with “-Disabled” if EVALUATE1.val is n
- Otherwise appended with “-Not set”

11.4.11. Macro evaluator

Tip: *Use the Action search to navigate Automate*

Overview

Automate provides a macro evaluator for testing macros at a hierarchy level.

Note: The macro evaluator is by default not available to the `sysadmin` administrator.

Refer to the macro functions, syntax and examples in the documentation.

On the **Macro Evaluator** page you can evaluate a macro, and also:

- Click the **Reset** button to revert the form to the default example.
- Click the **JSON Edit** button to edit your macro and context in a JSON editor.
- Use the **Copy to Clipboard** button to copy the macro output.
- Use the **Save to File** button to save the macro output to a JSON file - default filename is `Macro_Output.json`.

Evaluate a macro

1. In the Admin Portal, go to **Macro Evaluator**.
2. From the drop-down list, choose the **Context Hierarchy** at which the macro is to be evaluated.
3. Enter the macro text in the **Macro** input box.

   By default, a macro `{# data.Countries.* | country_name:input.country #}` is entered as a test macro to evaluate and **Show Context** is enabled to display the **Context Data** box for the `input.` parameter in the macro. Refer to the examples in the macro reference topics.
4. Enable and provide values for **Context Data** and add data if required.
5. Click **Evaluate** to run the query. The result of the evaluated macro is displayed in the **Output** box.

Index

M