



VOSS-4-UC Advanced Configuration Guide

Release 19.3.2

May 27, 2020

Legal Information

Please take careful note of the following legal notices:

- Copyright © 2020 VisionOSS Limited.
All rights reserved.
- VOSS, VisionOSS and VOSS-4-UC are trademarks of VisionOSS Limited.
- No part of this document may be reproduced or transmitted in any form without the prior written permission of VOSS.
- VOSS does not guarantee that this document is technically correct, complete, or that the product is free from minor flaws. VOSS endeavors to ensure that the information contained in this document is correct, whilst every effort is made to ensure the accuracy of such information, VOSS accepts no liability for any loss (however caused) sustained as a result of any error or omission in the same.
- This document is used entirely at the users own risk. VOSS cannot be held responsible or liable for any damage to property, loss of income, and or business disruption arising from the use of this document.
- The product capabilities described in this document and the actual capabilities of the product provided by VOSS are subject to change without notice.
- VOSS reserves the right to publish corrections to this document whenever VOSS deems it necessary.
- The terms Cisco, Movius, MeetingPlace, Netwise and all other vendor/product names mentioned in this document are registered trademarks and belong to their respective owners. VOSS does not own, nor is related to, these products and vendors. These terms have been included to showcase the potential of the VOSS solution and to simplify the deployment of these products with VOSS should you select to utilize them.

Security Information

This product may contain cryptographic features that may be subject to state and local country laws that govern the import, export, transfer and use of such features. The provision of this software does not imply that third-party authorization to import, export, distribute or use encryption in your particular region has been obtained. By using this product, you agree to comply with all applicable laws and regulations within your region of operation. If you require further assistance, please contact your dedicated VOSS support person.

Contents

- 1 What's New** **1**
 - 1.1 Advanced Configuration Guide: Release 19.3.2 1

- 2 Settings** **2**
 - 2.1 Login Banner 2
 - 2.2 Transaction Log Levels 2
 - 2.3 File Extensions 3
 - 2.4 Disable Device Logging 3
 - 2.5 Activate Phone Status Service 3
 - 2.6 Additional Role Access Profile Validation 5
 - 2.7 File Upload Limitations 6
 - 2.8 Time-To-Live for Uploaded Files 6
 - 2.9 Data Sync Blacklisted Attributes 6
 - 2.10 Global Settings 7
 - 2.11 Number Cooling Auto Expiry Schedule 9
 - 2.12 SMTP Server 9

- 3 System Monitoring** **10**
 - 3.1 System Monitoring Configuration 10
 - 3.2 System Monitoring Database Statistics 11
 - 3.3 System Monitoring Model Counts 12
 - 3.4 VOSS-4-UC Cluster Status 12
 - 3.5 UC Apps Reachability 13
 - 3.6 Worker Queue 14
 - 3.7 Login Sessions 14

- 4 Customization Overview** **16**
 - 4.1 Feature Package Customization 16
 - 4.2 GUI Customization 17
 - 4.3 Theme Customization 28
 - 4.4 On-line Help 40
 - 4.5 Scripts 42

- 5 Move Customizations (Provider)** **46**
 - 5.1 Network Device List Selection Rules Advanced Configuration 46

- 6 LDAP Sync** **49**
 - 6.1 Change LDAP User Sync from Top-Down to Bottom-Up 49

- 7 Custom Configuration** **55**
 - 7.1 Configuration Overview 55
 - 7.2 Site Defaults Reference 56
 - 7.3 Named Macros Overview 57

7.4	Quick Add Group Customization	57
7.5	Quick Add Subscriber Configuration	58
7.6	Smart Add Phone Configuration	63
8	Configuration Reference	66
8.1	Reference Material	66
8.2	Site Defaults Macros	66
8.3	Quick Add Groups Configuration Template Reference	72
8.4	Named Macro Reference	79
9	Macro Reference	85
9.1	Macros	85
9.2	Macro Syntax	85
9.3	Macro Functions	95
9.4	Macro Examples and Use	141
	Index	144

1 What's New

1.1. Advanced Configuration Guide: Release 19.3.2

- EKB-4053: UserOPS Workflows in QuickAddSubscriber and relationSubscriberPhone results in CUCM Purge of existing Phones. See: [Global Settings](#)
- EKB-4362: Update device/ldap/User blocked due to LDAP write-back failures on dSCorePropagation-Data and logonCount. See: [Data Sync Blacklisted Attributes](#)
- EKB-4642: Improve RIS API data collection performance. See: [Activate Phone Status Service](#)
- EKB-4666: Implement list operation to refresh status and IP address for a list of phones. See: [System Monitoring Configuration](#)
- EKB-5144: Update default Phone Status collection to be enabled. See: [Activate Phone Status Service](#)
- EKB-5144: Update default Phone Status collection to be enabled. See: [System Monitoring Configuration](#)

2 Settings

2.1. Login 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 - see [Set the Login Page Theme](#).

2.2. Transaction Log Levels

For users that have access to the `data/Settings` model, the levels of logging can be managed.

This level will affect the Log block of messages at the bottom of a selected transaction on the Transaction interface and does not for example affect the transaction or sub transaction Action and Detail information.

The levels includes messages as follows:

- Info - includes all messages
- Warning - includes Error and Warning messages
- Error - shows only error messages
- Disabled - disables all transaction log messages

2.3. File Extensions

For users with system administrator privileges to the `data/Settings` model, the valid file extensions can be managed.

By default, the following file extensions are in the Global instance of this model and are valid. This means that files with this extension can be uploaded to VOSS-4-UC:

- `.cer`
- `.crt`
- `.der`
- `.gzip`
- `.json`
- `.key`
- `.pem`
- `.xlsx`
- `.xml`
- `.zip`
- `.png`
- `.jpg`
- `.jpeg`

File extensions can be added or removed by this administrator. Files that do not have an extension that corresponds with an instance in this model, cannot be loaded or imported. An error message will display on the user interface to indicate that the file does not have a valid file extension.

2.4. Disable Device Logging

For users that have access to the `data/Settings` model, device logging can be managed.

If the Disable Device Logging check box is enabled, Unified CM AXL (and other external calls) requests and responses are not written to the transaction log. Generic driver requests, responses, template evaluations and macro evaluations are also not written to the transaction log.

These details will then *not* be shown for the relevant transactions under **Administration Tools > Transaction Log**.

2.5. Activate Phone Status Service

System administrators with access to `data/Settings` can see the **Activate Phone Status Service** check box that is enabled by default.

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. Carrying out this operation fetches the

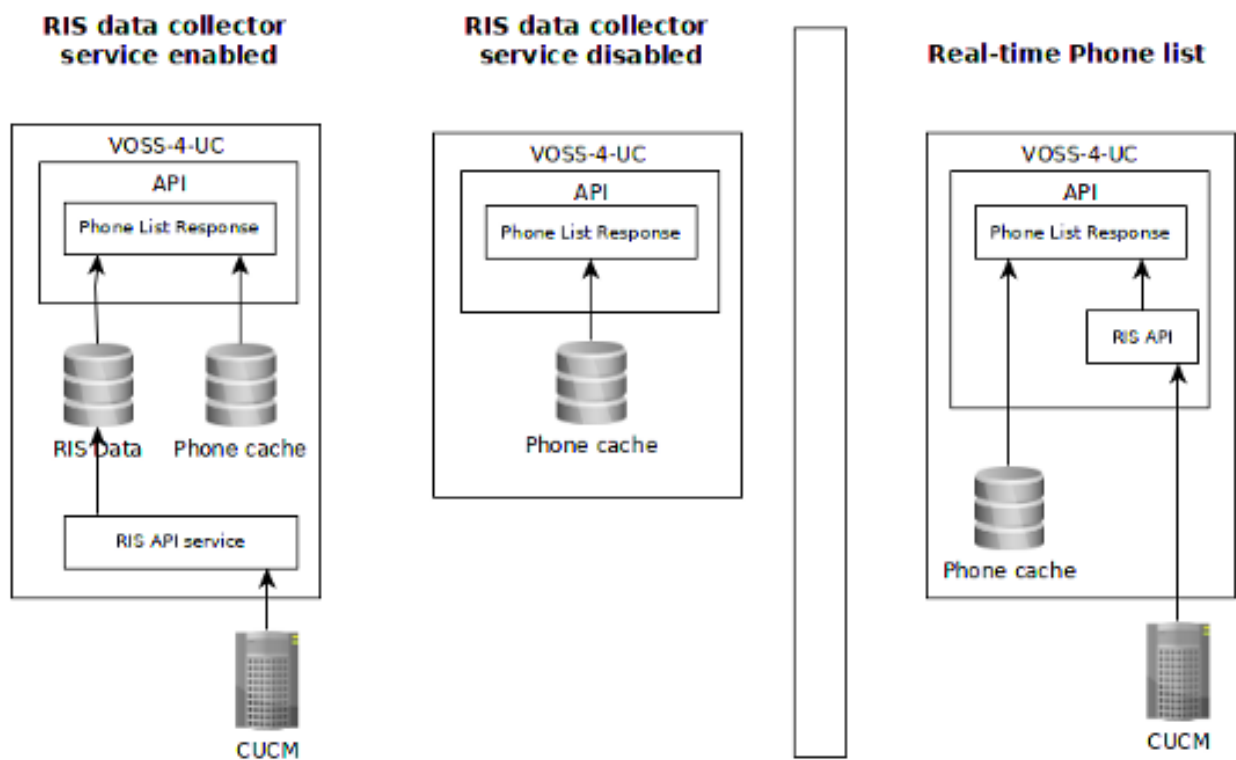
Unified CM phone IP address and status *directly* from the Unified CM and displays the data on the **Phones** list view **Registration Status** and **IP Address** columns, updating any existing data shown.

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.

As illustrated below, there is no cache of data resulting from this action. Any values shown in the columns 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 be updated 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
```

2.5.1. Phone Status Service in the Logs

When clearing this setting and then restarting the RIS API 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
}
```

2.6. Additional Role Access Profile Validation

For users with system administrator privileges to the `data/Settings` model, a setting called Additional Role Access Profile Validation is available as a check box to manage the available roles when an administrator creates Access Profiles.

- When the Additional Role Access Profile Validation check box is *enabled*:

An administrator 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 administrator'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.

- When the Additional Role Access Profile Validation check box is *disabled*:

An administrator can assign any role to a user, regardless of the administrator's 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.

By default, the Additional Role Access Profile Validation is disabled.

Refer to the macro topic Filter Role Functions for details on the use of the `fn.filter_roles_by_user_access_profile` function.

2.7. File Upload Limitations

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)

For users with permissions to the `Global` instance of the `data/Settings` datamodel, the upload file size and its time-to-live on the database can be configured.

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

- Bulk Load
- JSON import
- Theme upload
- any other file upload activity

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.

2.8. Time-To-Live for Uploaded Files

For users with system administrator privileges to the `data/Settings` model, the time-to-live for uploaded files can be managed.

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 Feature Guide.

2.9. Data Sync Blacklisted Attributes

Administrators with permissions to access the Global instance of the settings in the `data/Settings` model, can create a list of device attributes that will *not* trigger any update workflows that may have been defined to execute during the data sync. These attributes are here called “blacklisted” attributes.

The reason for blacklisted 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.

A number of blacklisted attributes have been added by default:

- For `device/cucm/User`:
 - `status`
 - `primaryDevice`
 - `attendeesAccessCode`

- displayName
- enableUserToHostConferenceNow
- pinCredentials
- passwordCredentials
- For device/cucm/Phone:
 - keyOrder
 - elinGroup
 - ecKeySize

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 blacklisted attributes instances will show as:

"Device changes on blacklisted attributes only. Updating cache, skipping workflows."

Note: After release 19.3.2 or applying patch EKB-4362-19.2.1_patch, the previously blacklisted LDAP attributes are no *longer imported* during LDAP synchronization:

For device/ldap/user:

- logonCount
- adminCount
- lastLogonTimestamp
- whenCreated
- uSNCreated
- badPasswordTime
- pwdLastSet
- lastLogon
- whenChanged
- badPwdCount
- accountExpires
- uSNChanged
- lastLogofflastLogoff

2.10. Global Settings

Provider administrators and higher have access to a **Global Settings** customization menu that allows for the configuration of a number of settings across all hierarchies or per individual hierarchy.

- On the GUI, the greyed-out value is the current setting either inherited from the hierarchy above or as a result of the selected drop-down option.
- The setting title and description provides a guide to its functionality.

Settings are enabled or disabled using a drop-down and can be one of 3 values:

- **Inherit:** the current setting is determined by the setting in the hierarchy above the current one. The greyed-out value is currently applied.
- **Yes:** enable the setting. This may change the current value.
- **No:** disable the setting. This may change the current value.

The following tabs are available:

- **Number Inventory**

- **Include the Number Inventory description in all number drop-downs:** when the Number Inventory is managed from the **Number Management** menu, descriptions can be added to the numbers (for example, “CEO number”). Default = **No**.

This setting controls the display where the number is listed on a form drop-down. The description will be displayed if the setting is **Yes**.

- **Enable Number Inventory Cooling:**

Default = **False** (not enabled).

Choose **Yes** from the drop-down to enable number cooling. If the inherited value is set to **True**, number cooling is already enabled.

See “Number Cooling” for more information.

- **Number Inventory Cooling Duration (Days):**

Default = 30 days.

If you want to choose a different value from the inherited value, then enter the required number of cooling duration days in this field.

- **Webex Teams**

- For the **Retain a Webex Teams User when a Subscriber is deleted** and **Send SNMP trap message when the Webex Teams Refresh Token expires** drop-downs, the values can be inherited or set as required.
- For the **Webex Teams Refresh Token expires threshold (in seconds)** drop-down, the default (inherited) value is 172800. In this case, the value itself can be changed, in other words there are no **Yes** or **No** options.

- **Email**

- For the **Allow email to be sent to user after Quick Add Subscriber:** By default the setting is inherited from the hierarchy level directly above the current one. When set to **Yes** at a hierarchy and a SMTP server is *also* set up for a hierarchy on the **Device Management** menu, a check box is available on the **Quick Add Subscriber** input form to send an email to the subscriber. See: [SMTP Server](#).

- **Phones**

- For **Delete existing Unassigned Phone when re-adding an identical Phone:** by default (inherited), the setting is **False**. This means that if a Phone with the same Name and Product Type is re-added to the system during for example a QuickAddSubscriber bulk load or re-add of a Subscriber during a Subscriber update, it will *not* by default be overwritten.

2.11. Number Cooling Auto Expiry Schedule

An `hcsadmin` user (or higher) has access to the **Number Cooling Auto Expiry Schedule (IniCoolingExpiryCleanupService)**, which can be managed from the **Scheduling** list view.

This schedule polls the **Cooling End Date** field on the **Number Inventory** list view on a daily basis to determine which numbers have completed their cooling period. These numbers are then returned to the pool of available numbers at the specific hierarchy.

Note: While the `hcsadmin` user can edit this schedule, for example change the execution time or timezone, the schedule **must run** every day, that is it **must not** be disabled.

2.12. SMTP Server

A SMTP server can be configured at a hierarchy to allow for email messages to be sent from VOSS-4-UC. This functionality is available in:

- Quick Add Subscriber - a check box becomes available on the input form if enabled in **Global Settings** on the **Email** tab. See: [Global Settings](#).
- File Transfer Destinations that can be configured by high level system administrators to transfer audit data for licensing. See the Advanced Configuration Guide.

Input the SMTP server details on the input form.

Note:

- Only one SMTP server can be set up at a hierarchy.
 - If a SMTP server is configured on a level above a hierarchy, this server will be used.
-

3 System Monitoring

3.1. System Monitoring Configuration

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**.

3.1.1. Alerts

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

- **Database Size Threshold (GB)**: the default size is 200 GB
- **Database Index Size Threshold (GB)**: the default size is 50 GB
- **Database Queue Size Threshold**: the default size is 500
- **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, SNMP traps are generated upon failure of the transaction. Refer to the System Monitoring SNMP topic in the Platform Guide for details and examples of the traps.

3.1.2. 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-4-UC 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 the PKID of the phone. For example:

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

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:

- 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
data/LdapUser
data/NormalizedUser
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. 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 or week 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 Business Admin Portal.

3.4. VOSS-4-UC Cluster Status

The `sysadmin` user has access to the **VOSS-4-UC Cluster Status** menu (menu model: `data/MonitoringCluster`) that shows aggregated VOSS-4-UC 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 shows. See the Cluster Check topic 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)
- **HCMF** (data/MonitoringHcmf)
- **LDAP** (data/MonitoringIdap)

The displayed data is the aggregated reachability and Round Trip Delay Time (RTT) for a UC app.

- 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 **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

The system makes use of components called *models* that can be created and modified. The primary types of models are:

- **Data Models** that are used to capture and store data.
- **Device Models** that represent components of available network devices.

A mechanism is available for higher level administrators to customize:

- how different attributes of a form are displayed (using Field Display Policies):
 - visibility (hidden / visible / read-only) of attributes
 - field names
 - related help text
 - ordering, grouping and layout
- how values for attributes are obtained (using Configuration Templates):
 - fixed / default values
 - derived from data in the system; optionally combined with input from users or Device Model events using pre-defined macros (see: [Macros](#))

Field Display Policies are applied to different roles using Menu Layouts and Landing Pages. Configuration Templates are applied to different roles using Menu Layouts, Landing Pages and pre-defined workflows.

The high level process that an administrator will follow is as follows:

1. Decide how an existing user input form should be customized. This involves an examination of the input form to identify field name, visibility, order and default values.
2. Determine the user who requires a modified input form.
3. Identify the User Role and Menu Layout associated with the relevant user.
4. If the menu item to be customized shows a Field Display Policy and Configuration Template associated with the item, clone these to start their customization. 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 Field Display Policy and Configuration Template according to identified requirements to create new instances of these. Note the uniqueness constrains per hierarchy on the names of the clones.

6. Associate the newly created Field Display Policy and Configuration Template 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

Field Display Policies are applied to certain item types in order to modify the default form that is displayed when these items are created or accessed.

With Field Display Policies, the fields on an item detail form can be grouped, disabled, and on-line help text can be added for a field. A field can be provided with a new label and its position on the form can be moved up or down.

More than one Field Display Policy can apply to a particular item type so that the selection of a particular policy will present another view of the form.

A Field Display Policy for an item type can be applied from a Menu Layout by selecting and associating it with the item on the Menu Layout. The Menu Layout is then selected to be part of a user Role so that users who have this role and log in will be able to have the item displayed according to the relevant Field Display Policy.

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 administration users at a certain level. Field Display Policies can then be made that are applied to the Menu Layout associated with the administration users at these levels.

A quick way to add a Field Display Policy is to clone an existing Field Display Policy, modify it as required and then to select it for the model on a user's menu layout. In this way a user's interface can be modified from the point of user access to the model on the menu.

There is a unique constraint on the name of the Field Display Policy per hierarchy level. The same name can be used on another hierarchy, but a new name is needed at the same hierarchy.

A Field Display Policy can also be selected for a form that is part of a Wizard.

If a Field Display Policy is called 'default', it will apply to a model by default.

4.2.2. Create a Field Display Policy

A Field Display Policy can be added to any of Data models, Relations and Views to modify the default form that is available for the item.

1. Choose the relevant hierarchy level.
2. Click **Add** from **Customizations > Field Display Policies**.
3. Enter the Name and optionally a Description for the Field Display Policy. If the name is 'default', the Field Display Policy will apply to the target model type by default.
4. Select or verify the model reference for the Target Model Type.
5. Click + adjacent to Groups and add a group Title. All the fields of the display policy should belong to a Group. If a group displays as a tab on the GUI, the Title is the tab title.

- Select the Display as Fieldset checkbox to show the group Title as a group header on the GUI and without tabs, or as a group title on a tab called Base if more than one group has this check box enabled.
 - Enter a numeric value for Number of Columns if the fields should show in columns.
6. On the Fields transfer boxes of the Group, select the required target model type fields from the Available box and add them to the Selected box. The available fields include fields from all versions of the schema of the target model type. Refer to the Rules When Creating a Field Display Policy topic for the caveats that apply when selecting the fields.
 7. For each selected field in a Selected transfer box, its attributes can be configured in the Field Overrides group:
 - Field - Select the field to which the attributes apply.
 - Title - Select the label text to be shown for the attribute.
 - Help Text - Enter text to display as the field on-line help and form tool tip for the attribute. If no text is added, the model attribute description is shown.
 - Disabled - select the check box if the field should show as disabled.
 - Input Type - The default display of the input field can be selected. For details, refer to the Field Display Policy Input Reference topic.
 8. Click **Save** on the button bar to save the Field Display Policy.

The created Field Display Policy is available to be applied to the model by using it in a Menu Layout.

4.2.3. Rules When Creating a Field Display Policy

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

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 below, 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

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. For 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. For 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. For 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. For 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. For 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. For 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. For 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. For example, to hide O.z.b, select O.z.a, O.w.d.

To order fields in a group, arrange them in the Selected box. Use the **Move Up** and the **Move Down** buttons under the box.

The following ordering rule applies:

The ordering of child fields and their parents depend 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 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.

4.2.4. FieldDisplayPolicy Field Reference

Title	Field Name	Description
Name *	name	The name that is given to the Field Display Policy.
Description	description	A description for the Field Display Policy instance.
Target Model Type *	target_model_type	The target model type to which the Field Display Policy applies.
Groups	groups	The groups that describe groupings of attributes that are displayed together on the user interface.
Title *	title	The name of a specific group of attributes.
Display as Fieldset	display_as_fieldset	Render this group as a fieldset in the form.
Number of Columns	num_cols	The number of columns of fields.
Policy *	policy	The specific policy application to attributes of the target model type of the Field Display Policy.

4.2.5. Clone a Field Display Policy

1. Login as provider administrator or higher.
2. Choose the desired hierarchy.
3. Choose **Customizations > Field Display Policies** to show the list of existing Field Display Policies.
4. Click on the field display policy that you want to clone.
5. Choose **Actions > Clone**.
6. Update the necessary fields for the cloned Field Display Policy. Refer to “Rules When Creating a Field Display Policy” for more information.
7. Click **Save**.

The cloned Field Display Policy is available to be applied to the item by selecting it in a Menu Layout that is available to a Role.

4.2.6. Field Display Policy Input Reference

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.

4.2.7. 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.

The screenshot shows the 'System Users' form with the following fields and their status:

- User Name**: Mandatory field (red box and asterisk).
- First Name**: Non-mandatory field.
- Last Name**: Non-mandatory field.
- Email Address**: Mandatory field (red box and asterisk).
- Password**: Non-mandatory field.
- Repeat Password**: Non-mandatory field.
- Role**: Mandatory field (red box and asterisk).
- User Authorization Method**: Dropdown menu with 'Standard' selected.

System Users		+
User Name	<input type="text"/>	*
Email Address	<input type="text"/>	*
Role	<input type="text"/>	*
First Name	<input type="text"/>	
Last Name	<input type="text"/>	
Password	<input type="text"/>	
Repeat Password	<input type="text"/>	
User Authorization Method	Standard	▼

4.2.8. Configuration Templates

Configuration templates are used to define values for attributes of any model. The values can be fixed values or existing macros visible from the hierarchy context where the configuration template is applied. The templates provide a useful way to define default values for items that are exposed in the GUI (visible, invisible or read-only). They also provides a mechanism to map data from data input via the GUI or device model events to other models or Provisioning Workflows in the system.

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

For example, if a model has an attribute that is defined to be a date string, a Configuration Template 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.

Another example is a model such as the Quick Add Subscriber that limits the user input to a few fields, whilst deriving the value of other hidden attributes from various Configuration Templates that are each applied to different underlying models that make up a Subscriber (for example, Voicemail account settings, conference account settings, phone, line, device profile settings, and so on).

When an instance of the model is added or updated, the Configuration Template that has been enabled for the model applies. 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.

More than one Configuration Template can be created for a model. These can then be used as needed. Configuration Templates 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 Configuration Template to a model that is selected as a menu item.

For administrators at Provider Administrator level or higher, a quick way to create a Configuration Template would be to open a similar template from for example the **Customizations > Configuration Templates** menu and to customize a clone of it.

Administrators at levels above the Site Administrator can also customize these templates, including Field Display Policies.

Note: In a multi-cluster environment, Configuration Templates that result in device model drop-down lists on the GUI, may contain duplicates. Any duplicated item can be selected by the user.

4.2.9. Create a Configuration Template

1. Choose the hierarchy level of the source model.
2. Open the menu item at its list view.
3. Click the **Configuration Template** button on the button bar.
4. Enter the Name and Description, and verify the Target Model Type.
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.

4.2.10. 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

4.2.11. 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.

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"
          }
        ]
      }
    }
  ]
}
```

4.2.12. 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. 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.
-

4.2.13. 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"
      }
    ]
  }
}
```

(continues on next page)

(continued from previous page)

```

    ],
    "type": "device/ios/Script",
    "name": "ConfigurationTemplateOverride"
  }
}

```

4.2.14. 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.

4.3. Theme Customization

4.3.1. Less files and Theme Customization

After downloading and extracting a theme into a directory, the contents shows the following directories and files:

```

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

```

Three files are available for theme customization:

- `variables.less`: custom variables overriding variables set elsewhere. To be compiled as `variables.css` when done.
- `skin.less`: main customization file to be compiled to `skin.css` when done.

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

The override hierarchy of variables and theme settings is seen as the first lines of `skin.less` as `@import` instructions, 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 those imports higher up on the list. The (optional) parameter indicates the import is ignored if the file is not available.

It is not advised to remove any references to files in the `build_dependencies` directory.

Customizing `variables.less`

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`. The variable names have self-explanatory names and are grouped into GUI categories, such as login page, landing page 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: 161px;
@loginLogoHeight: 51px;
```

These variables are used in `skin.less` and determine:

- 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 landing page logo image size HTML attributes:

```

```

Customizing `skin.less`

Customization is carried out by overriding the entries in the imported `.less` files present in the sub-directories of `build_dependencies`. In the compiled `.css` file, the override theme changes will then

follow the original CSS entries.

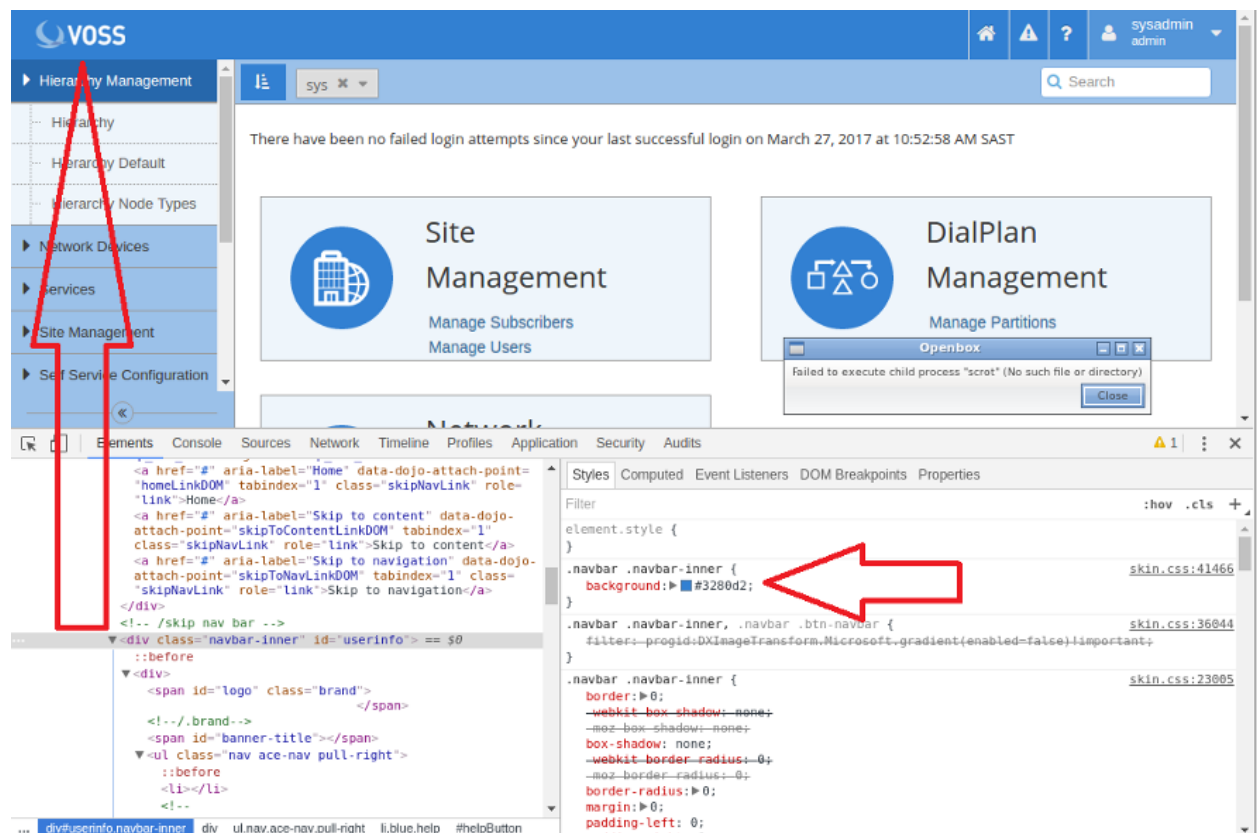
It is recommended that the browser developer tools be enabled in order to identify matching HTML properties of portions of the GUI, as well as the associated GUI styling in the `skin.css` file.

For 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;
}
```

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

The application of this override to the theme is shown by inspection in the browser developer tools:



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;
}
```

(continues on next page)

(continued from previous page)

```
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);
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='#ffffff',
    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.

Another example: if the alert notification pop-up style (for example with timeout notifications) should also be customized to match the custom style, the following style can be added - with the 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:

- [Download and Update a Theme](#)
- [Less files and Theme Customization](#)

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:

$$(\text{banner width} - \text{login form width}) / 2$$

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:

$$(800\text{px} - 400\text{px}) / 2 = -200\text{px}$$

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

```

h1.banner{
  width: 800px;
  margin-left: -200px;
  ...
}

```

4.3.3. Privacy Policy Menu Items

In order to comply with General Data Protection Regulation (GDPR) requirements, VOSS-4-UC provides the means to manage privacy policy notices on the user interface.

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 GUI, 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, 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 GUI, the Privacy Policy menu item is not visible from a menu layout and cannot be managed from **Role Management > Menu layouts**.

- Login page privacy policy links are managed from **Role Management > Themes**. Refer to [Set the Login Page Theme](#).

4.3.4. Manage Privacy Policy Menu Items

1. Log in as an administrator with the required privacy policy management permissions and menu access.
2. Choose the menu item, for example by default, **Role Management > 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 GUI, 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. Download and Update a Theme

1. Choose the hierarchy level at which the theme will be applied. Note that themes can only be customized by a Provider Administrator (or higher).
2. Choose **Role Management > Themes** to open the **Themes** list view.
3. Click the theme that you want to download.
4. Click **Action > Download** on the button bar to download the theme. The exported file is a .zip archive with the name of the theme. The archive contains a folder with the theme name and files called skin.css and skin.less in it.
5. Refer to the recommended practice to edit the Less file at the end of this topic, otherwise save the file and open the CSS file in a text editor.
 - Headers are clearly marked and apply to design areas within the GUI.
 - Colors, sizes, fonts, images, and so on, can all be overwritten with the required formats.
 - Images must be identified with the correct file path name. Preferably, do not use '/' preceding the path name; rather use relative paths, in other words a path relative to the CSS file location. For example, if your created an image sub-folder called 'img', use 'img/myimage.png'. After upload, the image should be viewable when opening the URL: *http://<hostname>/www/themes/mytheme/img/myimage.png*
6. When editing is complete, make sure the directory folder name is the same as your theme name. Compress the folder and save the file with the theme name and a .zip file extension.

An error message will display on the user interface if the file does not have a valid file extension.

Note that any files or folders inside the zip file archive that start with a '.' character will be silently discarded when unzipping the theme. For example, if the zip archive contains any files named `._.DS_Store` or `.directory`, they will be ignored.

7. Choose **Role Management > Themes**:

- a. For an update, choose the theme name and click **Browse** adjacent to **Import File** and then open the same theme name. Optionally select the **Backup Enabled** check box to create a backup of the current theme on the server. Click **Save** to complete the import process.

If a theme update does not require an updated CSS in a zip file but only updates the text of the theme, then no file upload is required. For information about the other fields on this form, refer to [Add a Theme](#).

- b. To delete a theme, choose it from the list and click **Action > Delete**.

The preferred way to edit a theme is to edit and compile the Less files. Refer to the “Less files and Theme Customization” topic in the “Advanced Configuration Guide”.

- All changes that are made directly in the CSS files will have to be manually carried over after each changes in the Less files.
- Edit the Less files and compile them to get the new CSS files.
- The aim with Less is that a theme can be customized with the minimum of technical knowledge (as with for example Twitter Bootstrap [<http://getbootstrap.com/customize/>]).

For an introduction to Less, visit the official website [<http://lesscss.org/>]. To compile the Less theme, a Less compiler is required. The following list shows examples:

- 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>]

4.3.6. Add a Theme

1. Prepare the theme file:

- a. Create a folder and add a file `skin.css` with the required name of the theme. The name of the folder must be the same as the intended theme name (only alphanumeric characters can be used, with no spaces or special characters).
- b. Add any CSS overrides to the file. Note that only the definitions as shown in the export of a provided CSS file `skin.css` can be modified.
- c. Add required image files in this folder (if any).
- d. Create a `.zip` archive file with the same filename as the folder.

Refer to [Download and Update a Theme](#) for details.

2. Add the theme to the system:

- a. Choose the hierarchy level at which the theme will be created.
- b. Choose **Role Management > Themes** to open the **Themes** list view.
- c. Click **Add** on the button bar to open the **Themes** input form. Note that themes can only be customized by a Provider Administrator (or higher).
- d. Enter the **Theme Name** (same as the file name created above).
- e. Enter an appropriate **Site Title** if required. The site title entered here is the title displayed in the browser tab.
- f. Click the **Browse** button to import the created theme zip file. Wait until the system displays the file chosen in the **Import File** field.

- g. If the theme must also apply to the login page, select the **Use this Theme to style Login page** check box.
- h. To set login banner text and notices on the login page, refer to [Set the Login Page Theme](#). The **Use this Theme to style Login page** check box does not have to be selected for banner text to show.
- i. If you want to hide the theme from lower level hierarchies, select the **Hide from Lower Hierarchies** check box. Users logged in at lower level hierarchies will not be able to see that particular theme on the **Themes** list view screen.
- j. For details on the **Theme Customization** tab controls, refer to [Create a Theme in the Business Admin Portal](#). .. note:

If *any* details are added on this tab, **all** fields should be provided **with** `↔input`.


- k. Click **Save** on the button bar when complete.

4.3.7. Create a Theme in the Business Admin Portal

The **Themes** interface can be used to create a theme that applies to the Business Admin Portal as well as the VOSS-4-UC Administration GUI.

Note:

- A theme created in the Business Admin Portal will also show and can be edited from the VOSS-4-UC Administration GUI (see under “Themes” in the Core Feature Guide for details).
- Since two themes are now compiled when saving a new theme on the Business Admin Portal, this takes slightly longer than theme compilation on the VOSS-4-UC Administration GUI.
- Themes created in the Business Admin Portal:
 - cannot be exported in full from the Business Admin Portal, so that it can be imported
 - will show in the VOSS-4-UC Administration GUI, where they can be exported

Use the **Preview** button  to preview the saved theme before it is applied to the user interface by assigning it to a user role.

Theme Details

Note: If the Business Admin Portal Theme interface is to be used for full Administration GUI theme configuration, a Custom Admin GUI File should be uploaded.

- **Custom Admin GUI File:**
 - If a *theme file is added*:
 - the theme file will apply the VOSS-4-UC Administration GUI, while the theme customization updates made in the **Branding** group of controls will *only* apply to the Business Admin Portal.

- If *no theme file is added*:

the theme updates made in the **Branding** group of controls are applied to VOSS-4-UC Administration GUI *and* the Business Admin Portal. The **Custom Admin GUI File** upload control is then *not* available for theme modification.

Branding

- Color selection:
 - Select a color using either the color picker control or by typing in the color hex value. The color picker control is only available on the Business Admin Portal theme interface.
 - Refer to:
 - [Business Admin Theme Color Reference](#).
 - [Administration GUI Theme Customization Color Reference](#)
- Image upload:
 - Note the file size and *width x height* pixel dimension size restrictions (shown on the context sensitive help) for images. A warning message will also show if selected images are too large.
 - The **Logo** image format must be PNG format. Other images can be PNG or JPEG.
 - The **Menu Background** image does not apply to the VOSS-4-UC Administration GUI.

4.3.8. Business Admin Theme Color Reference

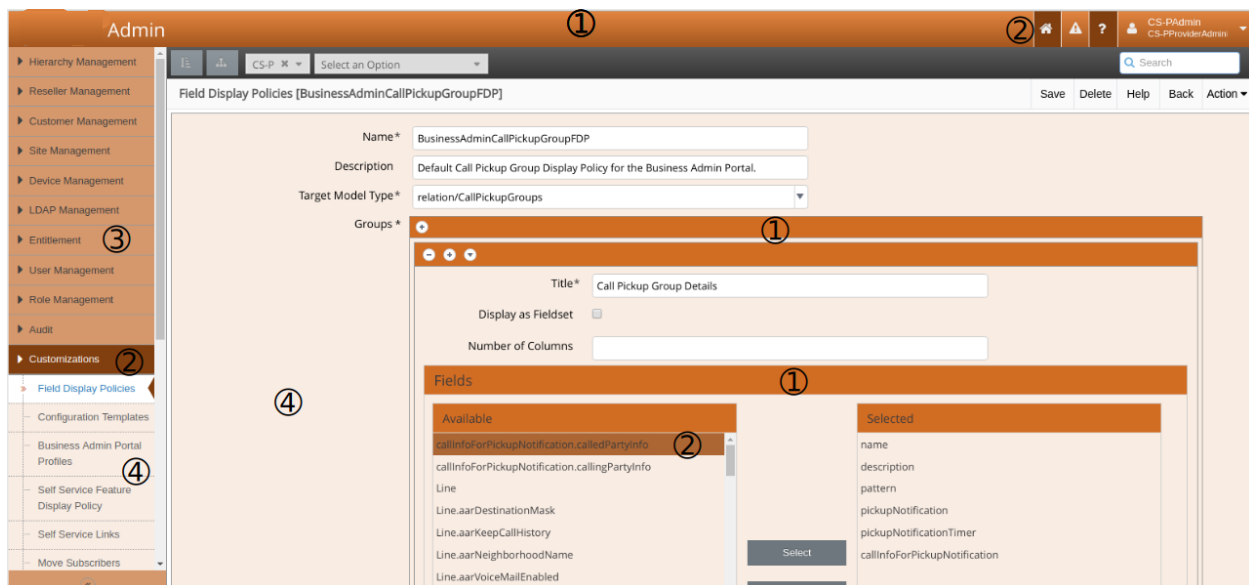
The Element variable below is a descriptive name of the GUI element that will be affected by the Color variable, which corresponds with the color name on the Theme design form.

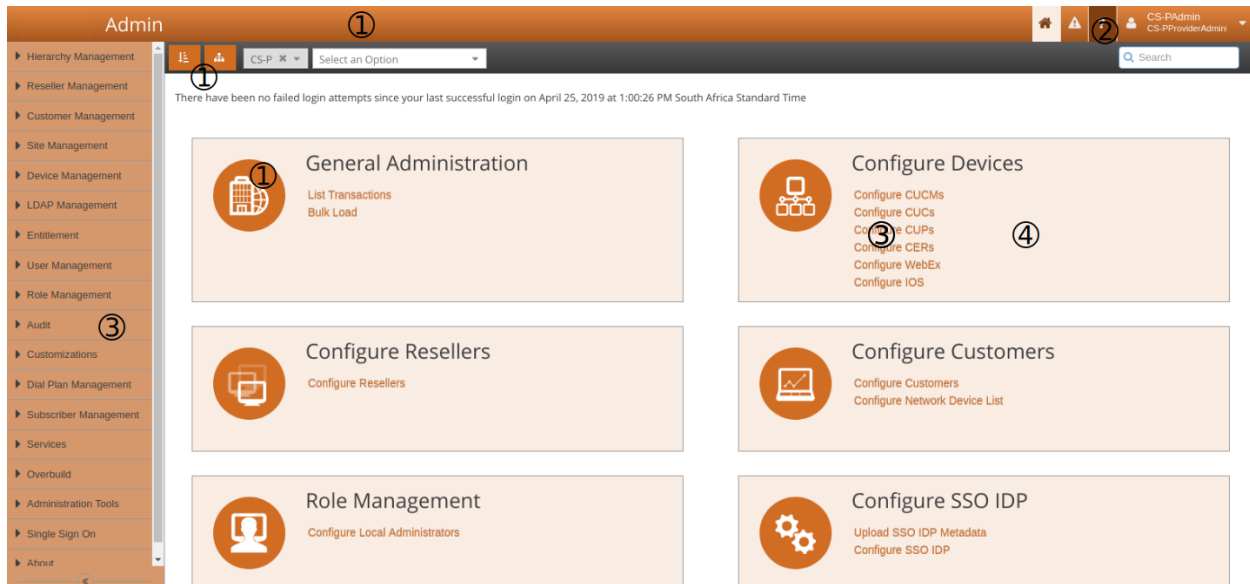
Element	Color
\$sidebarLogoBgColor	\$primaryColor
\$sidebarBgColor	\$primaryColor
\$topbarTextColor	\$primaryLightestColor
\$topbarIconColor	\$primaryLightestColor
\$submenuBgColor	\$primaryColor
\$darkSubmenuBgColor	\$primaryDarkColor
\$horizontalSubmenuBgColor	\$primaryColor
\$horizontalSubmenuItemHoverBgColor	\$primaryColor
\$horizontalSubmenuItemDarkHoverBgColor	\$primaryColor
\$menuItemActiveBgColor	\$accentColor
\$subMenuItemActiveTextColor	\$accentLightColor
\$subMenuItemActiveIconTextColor	\$accentLightColor
\$darkMenuItemActiveBgColor	\$accentColor
\$darksubMenuItemActiveTextColor	\$accentLightColor
\$darksubMenuItemActiveIconTextColor	\$accentLightColor

4.3.9. Administration GUI Theme Customization Color Reference

The Theme Customization Color names and the corresponding Administration GUI controls that are affected, are listed below:

1. Primary Colour
 - main window top bar
 - landing page icon background
 - hierarchy and tree view button
 - form heading bars
2. Dark Primary Colour
 - home and help buttons on main window top bar
 - selected side menu item
3. Light Primary Colour
 - hyperlink text
 - side menu background
4. Very Light Primary Colour
 - landing page card background
 - detail form background
 - sub menu background
5. Accent Colour
 - login button
6. Light Accent Colour





4.3.10. Set the Login Page Theme

1. Choose the hierarchy level in which the theme was created or to which the theme belongs.
2. Choose **Role Management > Themes** to open the Themes list view.
3. Click the required **Theme Name** that you want to use for the Login page.
4. On the **Base** tab, select the **Use this Theme to style Login page** check box.

If the check box is selected, and a theme with the same Interface already has the Login page check box selected, this will result in the disabling of the check box on the existing theme. There can only be one Interface-Login page combination on the system, so that any new themes or updates to existing themes may modify other themes on the system with the same Interface by disabling their Login page attributes. If no Interface is specified, the new login theme will also have its interface set to Administration.

5. On the **Login Page Details** tab, enter the required text in the **Title** field and **Banner Text** field. The text entered in the **Title** field will be displayed at the top of the Login page, above the logo. The text entered in the **Banner Text** field will be displayed at the bottom of the Login page, below the **Log In** button.

The banner text input box allows for multi-line input so that paragraphs can be created.

The maximum character length of the banner text is 2048 characters.

Banner text is displayed in the format that it is entered into the input box upon configuration. If cookie and security policy references are added, these show as links that open in new browser tabs.

6. In the **Banner Text** field, references to the cookie policy and privacy policy are added to the text as placeholders:

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

Input boxes are available on the Banner input form to add a caption and URL for each.

Although the cookie and privacy policy references are both optional, both their caption and URL fields become mandatory if they are used. The placeholder reference is then also required in the banner input box.

For suggested input on the use of cookies by VOSS-4-UC in the cookie policy text, see [VOSS-4-UC Cookie Policy](#).

Note that if a browser has a blocker installed that prevents new tabs from opening, this will affect the functionality of the links in the login banner.

Privacy policy links can also be added to user menus. Refer to [Privacy Policy Menu Items](#) and [Manage Privacy Policy Menu Items](#).

To customize the banner text style, refer to the topic on “Theme Banner Customization” in the “Advanced Configuration Guide”.

7. Click **Save** on the button bar when complete to implement the selection.

The Login page theme can also be applied to the login page during the log in process. Do this by adding the suffix ‘?theme=default’ to the login request url.

For example: `https://instance/login/?theme=default`,

where ‘default’ is one of the themes available in VOSS-4-UC. This usage in the URL will apply and override any theme that is set as the login theme.

4.3.11. VOSS-4-UC Cookie Policy

When formulating a cookie policy, customers should include details on the use of cookies by VOSS-4-UC. The text below provides details on the use of cookies in VOSS-4-UC that can be included in the policy:

VOSS-4-UC 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:

* Administrator login:

```
csrftoken
sessionid
```

* Self-service login:

```
csrftoken
sessionid
session
rbacInfo
```

4.3.12. Default Business Admin Subscriber Profile

If you don't want the **Default** Subscriber Profile to appear as an option when adding a Subscriber, a `sysadmin` user *only* can delete it from the **Subscriber Profiles** list view.

4.4. On-line Help

4.4.1. On-line Help Customization

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 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 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`.
- The instance name suffix, for example `_en-us` is replaced by the new language, for example `_fr-fr`. This name change is necessary because instance names should be unique.

If write access to the GeneralHelp model is available, 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 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.

4.4.2. Online Help Model Override Options

When adding or modifying an instance of the online help, the Model Override Options tab in the GUI 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.

4.5. Scripts

4.5.1. 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.

4.5.2. Create 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. Create 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.

4.5.3. 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}}
```

4.5.4. 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": {
```

(continues on next page)

(continued from previous page)

```

    "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 |
↵code |
      country_name:'South Africa'}}\nping\nping {{data.Countries.iso_country_
      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": {
      "pkid": "54dc76c72afa4327de0d217f",
      "bkey": "[\"10.120.2.175\", \"8443\", \"P.C\"]"
```

(continues on next page)

(continued from previous page)

```
    },  
    "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 from Top-Down to Bottom-Up

Top-down user LDAP user management means that LDAP users are first added to VOSS-4-UC and then synced to Unified CM. The steps below provide details on how to change LDAP user sync from top-down to bottom-up, in other words, LDAP users on Unified CM are synced to VOSS-4-UC.

Important: The precautions below should be taken before carrying out the change.

6.1.1. Preliminaries

- Take a VM snapshot before making any significant changes.
- Ensure that the LDAP server is in sync with VOSS-4-UC and that VOSS-4-UC is in sync with Unified CM.
- Make sure that you have the correct LDAP server information, or that someone is available who has the correct information.
- Make sure that Cisco and VOSS are aware of this change before commencing. L3 support staff need to be aware of the work being done beforehand.
- Always test the procedure for one user only first, using a Model Instance Filter. You need the assistance of VOSS-4-UC support
 - If the Model Instance Filter is to apply to the top down LDAP to VOSS-4-UC synced user, it should be on the `device/ldap/user` and the attribute `cn` - you can get the `cn` from the LDAP Synced users list.
 - If the Model Instance Filter is to apply to the bottom up, Unified CM to VOSS-4-UC synced user, it should be on the `device/cucm/user` and the attribute `userid`.

6.1.2. Checks

1. The Users list in VOSS-4-UC 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.

6.1. Change LDAP User Sync from Top-Down to Bottom-Up

Username	Given Name	Surname	Email Address	User Type	Department	Street	City	State	Postal Code	Building	Located At
TestLDAPAuth	Paul	TestLDAP...		VOSS-LDAP Synced	Admin						AAAGlobal (Customer)
tjankers01	Tim	Jankers	tjankers01@ab...	CUCM-LDAP Synced	Finance						AAAGlobal (Customer)
tjuliane01	Tim	Juliane	tjuliane01@ab...	CUCM-LDAP Synced	Engineering						LOC004 (Site)
tmbeteleste	mbele	test		VOSS-LDAP Synced							AAAGlobal (Customer)
User09h45		TheSurna...		VOSS Only							LOC004 (Site)

- The User Status column for the user in Unified CM is "Active LDAP synchronized User".

User ID	First Name	Last Name	Department	Directory URL	User Status
grantw	Grant	New			Active LDAP Synchronized User
grantw	GrantW	GrantW		grantw@vossqa.net	Active LDAP Synchronized User

- The LDAP server is configured on CUCM and that the LDAP Attribute for User ID is the same as the Login Attribute Name on VOSS-4-UC. (On Unified CM: **System > LDAP > Server** and **System > LDAP > LDAP Directory** and search to find it or add it.)

LDAP System Configuration

Status: Please Delete All LDAP Directories Before Making Changes on This Page

LDAP System Information

Enable Synchronizing from LDAP Server

LDAP Server Type: Microsoft Active Directory

LDAP Attribute for User ID: sAMAccountName

*- indicates required item.

LDAP Directory

Status: Ready

LDAP Directory Information

LDAP Configuration Name: 10.120.2.221

LDAP Manager Distinguished Name: cn=idap,cn=users,dc=vossqa,dc=net

LDAP Password: [Redacted]

Confirm Password: [Redacted]

LDAP User Search Base: ou=GRANTW,dc=vossqa,dc=net

LDAP Custom Filter: < None >

LDAP Directory Synchronization Schedule

Perform Sync Just Once:

Perform a Re-sync Every: 7 DAY

Next Re-sync Time (YYYY-MM-DD hh:mm): 2017-11-11 00:00

Standard User Fields To Be Synchronized

Cisco Unified Communications Manager User Fields	LDAP Attribute	Cisco Unified Communications Manager User Fields	LDAP Attribute
User ID	sAMAccountName	First Name	givenName
Middle Name	middleName	Last Name	sn
Manager ID	manager	Department	department
Phone Number	telephoneNumber	Mail ID	mail
Title	title	Home Number	homephone
Mobile Number	mobile	Pager Number	pager
Directory URL	msRTCSIP-primaryuseraddress		

- Confirm in the VOSS-4-UC schedules and transactions that recent LDAP - VOSS-4-UC syncs have

taken place and that Unified CM has the same user count as VOSS-4-UC.

5. Make sure in VOSS-4-UC that on **LDAP Management > LDAP User Sync** the user modes for Move, Delete and Purge are set to Manual. Note that when this configuration is saved, it will run a full LDAP sync.

6.1.3. Before you carry out the change

In VOSS-4-UC, make backups of LDAP server and configurations. The easiest way to do this 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**

This step is in case there are any issues. However, exporting is limited to 200 at a time, so for a customer with e.g. a 5K user count this is impractical. In that case a VM snapshot is recommended.

6.1.4. Make the change

1. In VOSS-4-UC, remove the instance under **LDAP Management > LDAP User Sync** for this customer.
2. Check that the users in question show as local users on both VOSS-4-UC (“CUCM Local”) and Unified CM (“Enabled Local User”).

Username	Given Name	Surname	Email Address	User Type	Department	Street	City	State	Postal Code	Building	Located At
user00021002	user0002	L002	user00021002...	CUCM Local							LOC002 (Site)
user00021003	user0002	L003	user00021003...	CUCM Local							LOC003 (Site)
user00021004	user0002	L004	user00021004...	CUCM Local							LOC004 (Site)
user00021005	user0002	L005	user00021005...	CUCM Local							LOC005 (Site)
User09h45		TheSurna...		VOSS Only							LOC004 (Site)

User ID	First Name	Last Name	Department	Directory URI	User Status
grantw@vossqa.net	Grant	New			Enabled Local User
grantw	GrantW	GrantW		grantw@vossqa.net	Enabled Local User

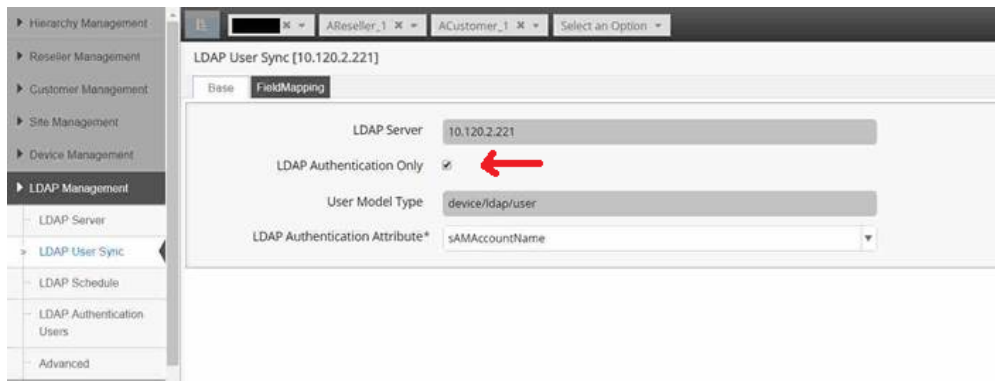
3. Enable the Cisco DirSync Service on Unified CM. Go to **Cisco Unified Serviceability Tools > Service Activation**. At the bottom of the page you will find Cisco DirSync Service. It will take some time to complete.

Service Name	Activation Status
Cisco DirSync	Activated

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



5. Check the user status of the user in Unified CM. The User Status will now show as “Active LDAP synchronized user”
6. In VOSS-4-UC, add the LDAP User Sync again and enable the LDAP Authentication Only option.



7. Run a DataSync from VOSS-4-UC with Unified CM. (I.e. the data sync with name that starts with “HcsPull”)

6.1.5. To change LDAP User Data Sync back to Top Down

1. Stop the DirSync service on Unified CM.

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**.

6.1. Change LDAP User Sync from Top-Down to Bottom-Up

Cisco Unified Serviceability
For Cisco Unified Communications Solutions

Navigation: Cisco Unified Serviceability Go

ccmadmin About Logout

Alarm Trace Tools Snmp CallHome Help

Control Center - Feature Services Related Links: Service Activation Go

Start Stop Restart Refresh Page

Status:
Ready

Select Server
Server* CUCM-10-6-SU1a--CUCM Voice/Video Go

Performance and Monitoring Services

Service Name	Status	Activation Status	Start Time	Up Time
Cisco Serviceability Reporter	Not Running	Deactivated		
Cisco CallManager SNMP Service	Not Running	Deactivated		

Directory Services

Service Name	Status	Activation Status	Start Time	Up Time
Cisco DirSync	Started	Activated	Sat Dec 9 14:48:38 2017	0 days 01:14:06

If this move is permanent, you can also delete the LDAP directory and stop the LDAP server Sync: disable the Enable Synchronizing from LDAP Server check box.

Status
Please Delete All LDAP Directories Before Making Changes on This Page

LDAP System Information

Enable Synchronizing from LDAP Server

LDAP Server Type: Microsoft Active Directory

LDAP Attribute for User ID: sAMAccountName

* - indicates required item.

2. Now check the user's User Status in Unified CM - it will now show "Enabled Local User".
3. In VOSS-4-UC, remove the Authenticate Only LDAP User sync.
4. In VOSS-4-UC, add an LDAP User Sync to do full LDAP syncs. (Or you can just import the JSON file exported earlier.)
5. Go to **User Management > Sync & Purge > LDAP Users** and run the sync users from LDAP (Unselect the Remove Log Messages).

Entitlement

User Management

Users

Provisioning Status

Sync & Purge

LDAP Users

LDAP Re-Provision

LDAP Users

Remove Log Messages

LDAP Action* Synchronize users from LDAP

6. Check user in Unified CM and in VOSS-4-UC. The user status should be:

- Unified CM: "LDAP Active Synced"
- VOSS-4-UC: "VOSS-LDAP Synced"

7 Custom Configuration

7.1. 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, VOSS-4-UC 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 Subscriber 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 subscribers 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 add Subscribers (Quick Add Subscriber)

Customization of resources may involve:

- changing values (optionally using macros)

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

7.2. 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 Subscriber, 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).

7.3. Named Macros Overview

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.

7.4. Quick Add Group Customization

Quick Add Groups provide a selection of service defaults for use in Subscriber management, in particular when the Quick Add Subscriber feature is used. More than one Quick Add Group can be available at a site

so that it can be selected during the Quick Add Subscriber workflow. Refer to the topic on Quick Add Groups under the Site level documentation for an overview.

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. The service defaults that are applied at the site level are stored in a Quick Add Group called “default” and serves as a template for the one created at the site. Administrators can view this template, but it cannot be modified.

The service defaults that are available in a Quick Add Group are stored as Configuration Templates to services.

Customization is carried out at the site level and can be done by either:

- creating new Quick Add Groups (and Configurations Templates)
- creating and selecting different Configuration Templates for use in an existing Quick Add Group.

7.5. Quick Add Subscriber Configuration

7.5.1. Quick Add Subscriber Provisioning Workflow Structure

The Quick Add Subscriber (QAS) feature uses a View model to define its schema that also ultimately defines:

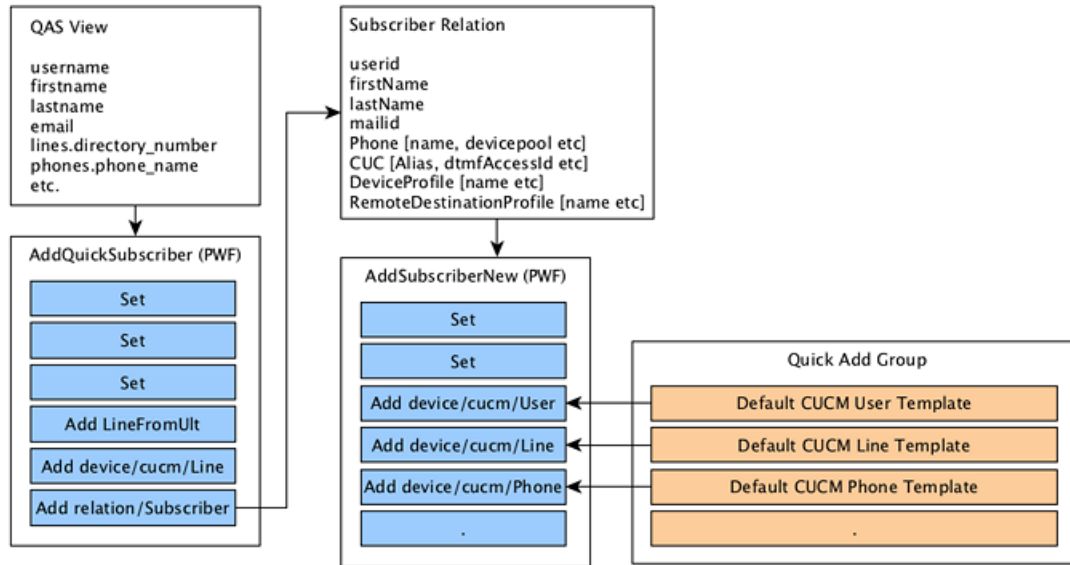
- The form fields a user sees in the GUI
- The column headings in the QAS Bulk Loader sheet
- The attributes supported by the API endpoint (`view/QuickSubscriber`)

This View schema represents a simplified version of all possible Subscriber fields.

Associated with the View is a Provisioning Workflow (PWF) called `AddQuickSubscriber` that captures these simplified fields and combines or maps them into the various components (services) of a full Subscriber. 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`, that separates out the data for each Subscriber 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 Config Templates of the selected Quick Add Group (QAG) is evaluated.



Since the Quick Add Group Config Templates (CFTs) are evaluated inside the `AddSubscriberNew` workflow, it is important that the Subscriber Relation field names *not* be those of the QAS 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 QAS View and the Subscriber Relation schemas, for example `username` as opposed to `userid`. The difference is subtle, but very important: the Subscriber user name in the QAS workflow context (`AddQuickSubscriber`) is `username`, but in the Subscriber Relation workflow context (`AddSubscriberNew`) it is `userid`.

7.5.2. Subscriber Relation Context

When adding a Subscriber, 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 Subscriber:

```
{
  "input": {
    "userid": "TestUserX77",
    "firstName": "Test",
    "lastName": "UserX77",
    "mailid": "testuserx77@mail.com",
    "Phone": [
      {
        "name": "SEP111222333444",
        "product": "Cisco Unified Client Services Framework",
        "protocol": "SIP",
        "lines": {
          "line": [
            {
              "dirn": {
                "pattern": "82007",

```

(continues on next page)

(continued from previous page)

```

        "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 will loop 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`.

7.5.3. Config Template Looping and Merging Overview

When the Quick Add Subscriber feature is used, the Subscriber 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 (QAG).

The purpose of the base template is to set fields which need to be hard coded to fulfill certain business rules, for example forcing the `ownerUserName` on Phone to the `userid` of the User.

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 which need to be looped through.

7.5.4. 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 screenshot shows the configuration for a CFT instance named 'Default CUCM Phone Template'. The 'Foreach Elements' section is expanded, showing the following configuration:

- Property***: `lines.line`
- Macro List***: `{# input.PhoneX.lines.line #}`
- Context Variable***: `LineX`

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 shows the configuration for a 'Line' instance. The 'Display Ascii' field is configured with the macro `{{ input.firstName }} - {{ cft.LineX.dirn.pattern }}`. The 'Enduser' field is currently empty, and the 'Ring Setting' is set to a dropdown menu.

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 Subscriber context, while the second macro references the pattern of the line instance being iterated over.

7.5.5. 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.

Description	Default CUCM Phone Template						
Foreach Elements	<div style="border: 1px solid #ccc; padding: 5px;"> <div style="background-color: #e6f2ff; padding: 2px; margin-bottom: 5px;">+</div> <div style="background-color: #e6f2ff; padding: 2px; margin-bottom: 5px;">- +</div> <table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">Property*</td> <td>lines.line</td> </tr> <tr> <td>Macro List*</td> <td>{# input.PhoneX.lines.line #}</td> </tr> <tr> <td>Context Variable*</td> <td>LineX</td> </tr> </table> </div>	Property*	lines.line	Macro List*	{# input.PhoneX.lines.line #}	Context Variable*	LineX
Property*	lines.line						
Macro List*	{# input.PhoneX.lines.line #}						
Context Variable*	LineX						
Schema Defaults	+						
Target Model Type*	device/cucm/Phone						
Merge Strategy	Replace						

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

7.6. Smart Add Phone Configuration

7.6.1. 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",

```

(continues on next page)

(continued from previous page)

```

    "macro_list": "{# input.lines #}"
  }
],

```

- Line specific customization for each line added on the Smart Add Phone GUI input form 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",
  "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 ))",

```

(continues on next page)

(continued from previous page)

```
"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"
    }
  ]
},
...
```

8 Configuration Reference

8.1. Reference Material

This section covers reference material for use in the customization of the Quick Add Subscriber 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 Subscriber Workflow Macro Reference

8.2. Site Defaults Macros

8.2.1. 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}}

8.2.2. 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}}

8.2.3. Subscriber 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 Subscriber Template
 - **Macro:** CUC_USER_templateAlias
 - **Default Value:** voicemailusertemplate

8.2.4. Quick 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

8.2.5. 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 Subscriber Template
 - **Macro:** CUC_HTML_NotificationTemplateName
 - **Default Value:** Default_Dynamic_Icons
- Default CUC HTML Notification Template
 - **Macro:** CUC_USER_templateAlias
 - **Default Value:** voicemailusertemplate

8.2.6. Webex Site Defaults

Configuration Template Applied: WebExUserPrePopulate

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

8.2.7. Hot Dial PLAR Site Defaults

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

8.2.8. Hunt Groups Site Defaults

Configuration Template applied to: `HuntGroupsPrePopulate`

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

8.2.9. Call Pickup Groups Site Defaults

Configuration Template applied to: `CPGPrePopulate`

- Default CUCM Call Pickup Partition
 - **Macro:** CUCM_CPUG_routePartitionName
 - **Default Value:** Site-{{macro.SITENAME}}

8.3. Quick Add Groups Configuration Template Reference

8.3.1. Default CUC User Template

applies to: `device/cuc/User`

Default CFT value:

- `defaultcucsubscribertemplate`: “voicemailusertemplate”

Value from a property of Site Defaults Doc for site, using a macro (line break added):

```

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

```

8.3.2. Default CUCM Phone Template

applies to: `device/cucm/Phone`

Internal workflow settings (not customizable):

- CFT has variables to loop over the lines on a Phone
- CFT replaces any existing values as opposed to only adding values

Default CFT values:

- `builtInBridgeStatus`: "Default"
- `callingSearchSpaceName`:

```
"(( input.Phone.callingSearchSpaceName == None ))
 <{{macro.CUCM_PHONE_callingSearchSpaceName}}>
 <{{input.Phone.callingSearchSpaceName}}>"
```

The CFT uses an IF-THEN-ELSE type macro to resolve the value. The macro first checks if a value for `callingSearchSpaceName` is available in the input context of the workflow (user input or for example by other means in the workflow such as another CFT). Otherwise, a named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultdevicecss`.

- `certificateOperation`: "No Pending Operation"
- `class`: "Phone"
- `commonPhoneConfigName`: "Standard Common Phone Profile"
- `description`: "Created by default template"
- `deviceMobilityMode`: "On"
- `devicePoolName`:

```
{{ macro.CUCM_PHONE_devicePoolName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultDP`.

- `lines.line[0].display`: "{{ input.firstName }}"

Each line associated to a phone will have the user first name that is available as input context to the workflow. The macro notation `input` is the namespace for the context reference that is available during the workflow. The value `firstName` is a property of the Subscriber object that is created and is available for the input form of the Quick Add Subscriber feature.

- `locationName`:

```
{{ macro.CUCM_PHONE_locationName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultLOC`.

- `packetCaptureMode`: "None"
- `phoneTemplateName`: "Standard 9971 SIP"
- `product`: "Cisco 9971"
- `protocol`: "SIP"

- protocolSide: “User”
- securityProfileName: “Cisco 9971 - Standard SIP Non-Secure Profile”
- useTrustedRelayPoint: “Default”

8.3.3. 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 }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of defaultDP.

- product: “Remote Destination Profile”
- protocol: “Remote Destination”
- protocolSide: “User”

8.3.4. Default CUCM RD Template

applies to: device/cucm/RemoteDestination

Default CFT values:

- answerTooLateTimer: “19000”
- answerTooSoonTimer: “1500”
- delayBeforeRingingCell: “4000”

8.3.5. Default CUCM User Template

applies to: device/cucm/User

Default CFT values:

- enableMobility: “((True))”
The value ((True)) is a macro in the system.
- presenceGroupName: “Standard Presence group”

8.3.6. Default Webex User Template

applies to: `device/webex/User`

- `description`: "AddWebEx"

By default, the template has no customization.

8.3.7. 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"

8.3.8. 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}}>"
```

The CFT uses an IF-THEN-ELSE type macro to resolve the value. The macro first checks if a value for `callingSearchSpaceName` is available in the input context of the workflow (user input or for example by other means in the workflow such as another CFT). Otherwise, a named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultdevicecss`.

- `certificateOperation`: "No Pending Operation"
- `class`: "Phone"
- `commonPhoneConfigName`: "Standard Common Phone Profile"
- `description`: "Created by default template"
- `deviceMobilityMode`: "Default"
- `devicePoolName`:

```
{{ macro.CUCM_PHONE_devicePoolName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultDP`.

- `locationName:`

```
{{ macro.CUCM_PHONE_locationName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultLOC`.

- `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"

8.3.9. 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}}>"
```

The CFT uses an IF-THEN-ELSE type macro to resolve the value. The macro first checks if a value for `callingSearchSpaceName` is available in the input context of the workflow (user input or for example by other means in the workflow such as another CFT). Otherwise, a named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultdevicecss`.

- `certificateOperation:` "No Pending Operation"
- `class:` "Phone",
- `commonPhoneConfigName:` "Standard Common Phone Profile",
- `description:` "Created by default template",
- `deviceMobilityMode:` "Default",
- `devicePoolName:`

```
{{ macro.CUCM_PHONE_devicePoolName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultDP`.

- `locationName:`


```
{{ macro.CUCM_PHONE_locationName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultLOC`.

- `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”

8.3.10. 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}}>"
```

The CFT uses an IF-THEN-ELSE type macro to resolve the value. The macro first checks if a value for `callingSearchSpaceName` is available in the input context of the workflow (user input or for example by other means in the workflow such as another CFT). Otherwise, a named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultdevicecss`.

- `certificateOperation`: “No Pending Operation”
- `class`: “Phone”
- `commonPhoneConfigName`: “Standard Common Phone Profile”
- `description`: “Created by default template”
- `deviceMobilityMode`: “On”
- `devicePoolName`:

```
{{ macro.CUCM_PHONE_devicePoolName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultDP`.

- `locationName`:

```
{{ macro.CUCM_PHONE_locationName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultLOC`.

- `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"

8.3.11. Default CUCM Line Template

applies to: `device/cucm/Line`

Default CFT values:

- `callingSearchSpaceName`:

```
{{ macro.CUCM_LINE_shareLineAppearanceCssName }}
```

- `routePartitionName`:

```
{{ macro.CUCM_LINE_routePartitionName }}
```

- `shareLineAppearanceCssName`:

```
{{ macro.CUCM_LINE_shareLineAppearanceCssName }}
```

8.3.12. 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}}>"
```

The CFT uses an IF-THEN-ELSE type macro to resolve the value. The macro first checks if a value for `callingSearchSpaceName` is available in the input context of the workflow (user input or for example by other means in the workflow such as another CFT). Otherwise, a named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultdevicecss`.

- `certificateOperation`: "No Pending Operation"

- class: “Phone”
- commonPhoneConfigName: “Standard Common Phone Profile”
- description: “Created by default template”
- deviceMobilityMode: “Default”
- devicePoolName:

```
{{ macro.CUCM_PHONE_devicePoolName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultDP`.

- locationName:

```
{{ macro.CUCM_PHONE_locationName }}
```

A named macro is used to resolve the value. The named macro references the value from the Site Defaults Doc at the current hierarchy, in particular, from the value of `defaultLOC`.

- 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”

8.4. Named Macro Reference

8.4.1. 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.

8.4.2. 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 }}
```

8.4.3. 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 }}
```

8.4.4. Named Macros - Site Defaults Line

- **macro.CUCM_LINE_presenceGroupName**

```
{{ data.SiteDefaultsDoc.default_cucm_line_presencegroup }}
```
- **macro.CUCM_LINE_vmprofile**

```
{{ data.SiteDefaultsDoc.default_cucm_line_vmprofile }}
```
- **macro.CUCM_LINE_routePartitionName**

```
{{ data.SiteDefaultsDoc.defaultlinept }}
```
- **macro.CUCM_PHONE_subscribeCallingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecss }}
```

- **macro.CUCM_LINE_callForwardNoAnswer_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfnacss }}
```

- **macro.CUCM_LINE_callForwardAll_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfacss }}
```

- **macro.CUCM_LINE_callForwardNoAnswerInt_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfnaicss }}
```

- **macro.CUCM_LINE_callForwardBusy_callingSearchSpaceName-2**

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

By default, `macro.DPSITE` resolves to the site name at the current hierarchy. This macro has the source: `{{ data.BaseSiteDAT.SiteName }}`

The macro therefore resolves to the Call Forward Busy CSS Name with the same name as the current site name.

- **macro.CUCM_LINE_callForwardBusyInt_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfbicss }}
```

- **macro.CUCM_LINE_callForwardNoCoverage_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfnccss }}
```

- **macro.CUCM_LINE_callForwardNoCoverageInt_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfncicss }}
```

- **macro.CUCM_LINE_callForwardOnFailure_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfofcss }}
```

- **macro.CUCM_LINE_callForwardNotRegisteredInt_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfnricss }}
```

- **macro.CUCM_LINE_callForwardNotRegistered_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfnrcss }}
```

- **macro.CUCM_LINE_callForwardAlternateParty_callingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlinecfapcss }}
```

- **macro.CUCM_LINE_callForwardAll_secondaryCallingSearchSpaceName**

```
{{ data.SiteDefaultsDoc.defaultlineseccss }}
```

- **macro.CUCM_LINE_shareLineAppearanceCssName**

```
{{ data.SiteDefaultsDoc.defaultlinecss }}
```

8.4.5. 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 }}
```

8.4.6. 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 }}
```

8.4.7. Named Macros - Site Defaults Hotdial

- **macro.CUCM_DP_defaultplarcss**

```
{{ data.SiteDefaultsDoc.defaultplarcss }}
```

- **macro.CUCM_HOTDIAL_defaultplarcss**

```
{{ data.SiteDefaultsDoc.defaultplarcss }}
```

8.4.8. 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 Subscriber - 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.

9 Macro Reference

9.1. Macros

Macros are used to return data from the system in various formats, to test for conditions, map data from GUI or bulk loader input to various elements in the system (in conjunction with configuration ltemplates) and to access data in workflow steps.

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.

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.

9.2. Macro Syntax

9.2.1. 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.

- { % 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 OR operator in a test is |, for example:

```
(( device.cuc.PagerDevice.Undeletable| ObjectId:input.ObjectId|
direction:local == False ))
```

- ((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.

9.2.2. Macro Syntax Multiline

Macros can contain line breaks. The GUI 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.

9.2.3. Dot notation

A dot-notation is used to refer to a macro function, model attribute, a defined variable, step reference, or non-attribute value in the model schema.

- *fn.macronym_name* - identifies a macro function.
- *self.attribute* is used to refer to the current value of an attribute in the model where the macro applies. Here, *attribute* should be an attribute name of the calling model in which the macro is referenced (usually a configuration template). In other words, the macro should be associated with a configuration template of a resource that is referenced.
- *previous.attribute* is used to the previously saved value of an attribute as opposed to the existing value in the case where a model is updated.
- *input.attribute* identifies the values input via any of: the GUI, bulk load, provisioning workflow foreach variable or context.
- *pwf.variablename* identifies a variable value defined as a provisioning workflow set step variable.
- *cft.attribute* identifies the values input in a configuration template via the current value of the *context_var* of a *foreach* loop of a configuration template.
- *modelname.attribute* - this notation defaults to the Data Model type.
- *modeltype.modelname.attribute* is used for other non-data model types.
- *modeltype.modelname.attribute.NUMBER* is used to refer to attribute NUMBER-1 where there is more than one attribute, that is, the first attribute reference is *modeltype.modelname.attribute.0*. See the Macro examples section. The NUMBER can also be a wild card, as in *{# device.cucm.Phone.lines.line.*.dirn #}*
- *macro.name* is used to refer to a defined macro by name.
- *workflow.stepSTEPNUMBER.pkid* - a hierarchy value to override the **Context Hierarchy** by specifying the context using the pkid of stepSTEPNUMBER, where STEPNUMBER can be 1, 2 and so on.

- Non-attribute notation allows the following for a model:

- `__pkid`
- `__bkey`
- `__hierarchy`
- `__hierarchy_friendly_path`
- `__hierarchy_friendly_parent_path`

Some examples of this syntax:

```
{# 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 }}
```

The hierarchy fields can be combined with model attribute names in a comma separated, 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"
  },
  ...
]
```

Meta attribute properties can also be used **in** a macro **filter**. For details **and** examples, refer to the topic on Macro Syntax to Filter by Meta Properties.

- 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}}
```

9.2.4. 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, the SELECT FROM notation can contain a wild card asterisk `*` to return all matching values.

Examples returning all User attributes:

```
{# User.* |username: js54321, last_name: 'van Dever' #}
{% 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 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
 - __hierarchy_friendly_path
 - __hierarchy_friendly_parent_path

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]
- | `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.

It is recommended to use a *direction* option in a list macro with out 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.

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 # }
```

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]
```

9.2.5. 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))
```

9.2.6. 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
```

9.3. Macro Functions

9.3.1. 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.

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 }}
```

9.3.2. Numeric Functions

- *fn.zeropad*: Left pad a given number with zeros up to a given pad number.
- *fn.minval*: For integers, return the minimum value of a provided list.
- *fn.maxval*: For integers, return the maximum value of a provided list.

- *fn.add*: Add two integers.
- *fn.subtract*: Subtract two integers.
- *fn.multiply*: Multiply two integers.
- *fn.divide*: Divide two integers.

Examples:

Example	Output
<code>{{fn.zeropad 123, 6}}</code>	000123
<code>{{fn.minval 2, 3, 130, 1, 30}}</code>	1
<code>{{fn.maxval 2, 3, 130, 1, 30}}</code>	130
<code>{{fn.add 2, 3}}</code>	5
<code>{{fn.subtract 2, 3}}</code>	-1
<code>{{fn.multiply 2, 3}}</code>	6
<code>{{fn.divide 20, 10}}</code>	2

9.3.3. String Functions

- *fn.index* : Return the *i*'th item of an iterable, such as a list or string.
- *fn.mask* : Return a mask of (length + modifier) instances of char.
- *fn.length* : Return the length of a string.
- *fn.split* : Split a string by delimiter, returning a list.
- *fn.join* : Join a string by delimiter. If no delimiter is provided, the list is returned as a single string.
- *fn.title* : Return a string in title case.
- *fn.upper* : Return an uppercase version of the input string.
- *fn.lower* : Return a lower case version of the input string.
- *fn.contains* : Return true or false if string is contained in another.
- *fn.sub_string* : Return the substring of a string from a start to an end position.

- *fn.containsIgnoreCase* : Return true or false if string is contained in upper- or lower case.
- *fn.containsStartsWith* : Return true or false if source string is the start of target string.
- *fn.containsStartOf* : Return true or false if start of source string is target string.
- *fn.isexactly* : Return true or false if source string is exactly the same as target string.
- *fn.replace* : Replace target substring for source substring in source string.
- *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.

Examples:

Example	Output
<code>{{ fn.index 'foo bar baz', 5 }}</code>	'b'
<code>{{ fn.mask X, 2, 3 }}</code>	XXXXXX
<code>{{ fn.length This is a valid string }}</code>	22
<code>{# fn.split foo:bar:baz,: #}</code>	['foo', 'bar', 'baz']
<code>{{ fn.join 1234, : }}</code> <code>{{ fn.join 1234 }}</code>	1:2:3:4 1234
<code>{{ fn.title 'foo bar baz' }}</code>	'Foo Bar Baz'
<code>{{ fn.upper somevalue }}</code>	SOMEVALUE

Example	Output
<code>{{ fn.lower SOMEVALUE }}</code>	somevalue
<code>{{ fn.contains needle, haystack }}</code>	false
<code>{{ fn.contains hay, haystack }}</code>	true
<code>((fn.contains Kit, 1234 Kit Creek == True))</code>	true
<code>{{ fn.sub_string haystack, 0, 2 }}</code>	hay
<code>{{ fn.sub_string haystack, 7, 7 }}</code>	k
<code>{{ fn.containsIgnoreCase aaa, bbbaAacc }}</code>	true
<code>{{ fn.containsStartsWith aaa, aaaffccffdd }}</code>	true
<code>{{ fn.containsStartOf ffnnccgg, ffnn }}</code>	true
<code>{{ fn.isexactly source1, source1 }}</code>	true
Note: no spaces between commas. <code>{{ fn.replace ddddAAAc,AAA,FFF }}</code>	ddddFFFc
<code>{{ fn.validate_name }}</code>	false
<code>{{ fn.validate_name a }}</code>	true

- *fn.fix_non_ascii* : Given a string containing non-ASCII characters and a “from” and “to” characters parameter pair, return a string with “from” characters replaced with “to” characters. This allows for control over the replacement.

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.
2. Map all “accented” (accent, circumflex, diaeresis, etc.) non-ASCII characters to ASCII characters.
3. Map all remaining non-ASCII characters with `_`.

For example:

– `string = Test Nón Ascii`

- `<from>,<to> = ï,G`
- `result = _est Non AsciiG`

step 1: `from = ï` is replaced with `to = G`.

step 2: accented `ó` is replaced with `o`

step 3: `T` is a remaining non-ASCII character and replaced with `_`.

Example	Output
<pre>{{ fn.fix_non_ascii 'Têst Nón Asciiæ mapping' ,sæ, →Ga }}</pre> <pre>{{ fn.fix_non_ascii 'Test Nón Ascii' }}</pre>	<pre>'TeGt Non AGciiæ mapping'</pre> <pre>'_est Non Ascii'</pre>

- `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
<pre>{{ fn.fix_username O'Reilly }}</pre>	<pre>O-Reilly</pre>

9.3.4. 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*
- `fn.list_index`: Return a specified item from a list. Zero is the first item.
- `fn.list_index_item`: Return the position of item in list.
- `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.
- `fn.list_count_item`: Return the number of times item is in a list.
- `fn.list_in`: Return true or false if item is in a list or not.
- **`fn.list_contain`: Return true or false if item is a substring of an input context** *or* is in a list or not.
- `fn.list_append`: Returns a list with item appended.
- `fn.list_pop`: Return the last item of the list.
- `fn.list_insert`: Return a list with item inserted at position.

- *fn.list_insert_no_dup*: Return a list with item inserted at position and all duplicates ignored.
- *fn.list_remove*: Return a list with all instances of item or list of items removed.
- *fn.list_remove_dup*: Return a list with all instances of item or list of items removed, including duplicates.
- *fn.list_remove_nulls*: Return a list with all `null` instances in a list of items removed.
- *fn.list_reverse*: Return a list that is the reverse of a given list.
- *fn.list_extend*: Return a list that is an extension of list 1 with list 2.
- *fn.list_extend_no_dup*: Return the concatenation of list1 and list2, ignoring duplicates.
- *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.
- *fn.list_set_symdiff*: Given two lists as input, return the list of items that are *not* common to both lists.
- *fn.list_sort*: Return a sorted list; by ascending (A) or descending (D) order.
- *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`.
- *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.
- *fn.list_empty*: Returns an empty list
- *fn.list_set_intersect*: Given two lists, return the intersection as a list.
- *fn.list_set_union*: Given two lists, return the union as a list.
- *fn.list_set_left*: Given two lists, return a list of items in the left list only.
- *fn.list_set_right*: Given two lists, return a list of items in the right list only.
- *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.
- *fn.flatten_list_of_lists*: Given a list of lists, return a single, flattened list.
- *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.
- *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
<p>Context hierarchy: <code>sys</code>, list of more than 1 of <code>iso_country_code</code> in <code>data/Countries</code>, search down</p> <ul style="list-style-type: none"> display as dictionary: <pre data-bbox="207 352 906 436">{{ fn.group_by_larger_than_count data/Countries, iso_country_code down, 1, dict }}</pre> <ul style="list-style-type: none"> display as list: <pre data-bbox="207 493 906 577">{{ fn.group_by_larger_than_count data/Countries, iso_country_code down, 1, list }}</pre>	<pre data-bbox="1221 283 1435 829">[{ "iso_country_ ←code": "AUS", "count": 2 }, { "iso_country_ ←code": "CHN", "count": 2 }] ["AUS", "CHN"]</pre>
<pre data-bbox="207 907 889 938">{{fn.list_index 2, data.Countries.country_name}}</pre>	<p>“Canada” if this is the third item.</p>
<pre data-bbox="207 1020 776 1104">MACRO1={# fn.sequence 40, 43 #} {{ fn.list_index_item 42,macro.MACRO1 }}</pre>	<p>2</p>
<pre data-bbox="207 1180 990 1379">{{fn.list_count data.Countries}} {{fn.list_count input.does_not_exist}} {{fn.list_count non_existant_namespace.does_not_exist}}</pre>	<p>25 if the list has 25 items.</p> <p>0</p> <p>1</p>
<pre data-bbox="207 1453 1003 1568">MACRO1={# data.Countries.international_access _prefix #} {{ fn.list_index_item 00,macro.MACRO1 }}</pre>	<p>19</p>
<pre data-bbox="207 1642 961 1757">{{fn.list_in 'AUS', data.Countries.iso_country_code}} {{fn.list_in 'AUZ', data.Countries.iso_country_code}}</pre>	<p>true</p> <p>false</p>

Continued on next page

Table 1 -- continued from previous page

Example	Output
<pre>{ "input": { "list": ["AUS", "BHR"], "key": "aaAUza" } } {{fn.list_contain 'AUS', input.list }} {{fn.list_contain 'AUZ', input.key }}</pre>	<pre>true true</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} {{fn.list_append 999, macro.MACRO1 }}</pre>	<pre>['40', '41', '42', '43', '999']</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} {{fn.list_pop macro.MACRO1}}</pre>	<pre>"43"</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} {{fn.list_insert 1, 999, macro.MACRO1 }}</pre>	<pre>['40', '999', '41', '42', '43']</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO7={# fn.sequence 39, 41 #} {{fn.list_insert_no_dup macro.MACRO1, macro.MACRO7}}</pre>	<pre>['40', '41', '42', '43', '39']</pre>
<pre>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 }}</pre>	<pre>['40', '41', '42'] ['42', '43']</pre>

Continued on next page

Table 1 -- continued from previous page

Example	Output
<pre> {"pwf": { "my_list": [1, null, 2, "test", 3, null] } } {{ fn.list_remove_nulls pwf.my_list }} </pre>	<pre>[1,2,"test",3]</pre>
<p>Given the following list is the result of the regex for iso_country_code:</p> <pre> {# data.Countries.iso_country_code iso_country_code:/AU/ #} {# fn.list_remove_dup data.Countries.iso_country_code iso_country_code:/AU/ #} </pre>	<pre> ['AUS','SAU', 'AUS','AUS'] ['AUS','SAU'] </pre>
<pre> MACRO1={# fn.sequence 40, 43 #} {{fn.list_reverse macro.MACRO1}} </pre>	<pre> ['43','42', '41','40'] </pre>
<pre> MACRO1={# fn.sequence 40, 43 #} MACRO9={# fn.sequence 50, 52 #} {{fn.list_extend macro.MACRO1,macro.MACRO9 }} </pre>	<pre> ['40', '41', '42', '43', '50', '51', '52'] </pre>
<pre> MACRO1={# fn.sequence 40, 43 #} MACRO8={# fn.sequence 42, 45 #} {{fn.list_extend_no_dup macro.MACRO1, macro.MACRO8 }} </pre>	<pre> ['40','41', '42','43', '44','45'] </pre>
<pre> {# fn.sequence 40, 43 #} {# fn.sequence 110, 100, 4 #} </pre>	<pre> ['40','41', '42','43'] ['0110','0109', '0108', '0107','0106', '0105','0104', '0103','0102', '0101','0100'] </pre>

Continued on next page

Table 1 -- continued from previous page

Example	Output
<pre>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}}</pre>	<pre>['43', '42', '41', '40'] ['43', '42', '41', '40'] ['0110', '0109', '0108'] ['0108', '0109', '0110']</pre>
<pre>{{fn.one data.Countries.iso_country_code emergency_access_prefix:'112'}} {{fn.one abc}}</pre>	<pre>A single result, e.g. 'FRA' 'abc'</pre>
<pre>{{fn.as_list data.Countries.country_name country_name:'China'}} {{fn.as_list abc}}</pre>	<pre>['China'] ['abc']</pre>
<pre>{{fn.list_empty}}</pre>	<pre>[]</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO2={# fn.sequence 41, 42 #} {# fn.list_set_intersect macro.MACRO1, macro.MACRO2 #}</pre>	<pre>['41', '42']</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO2={# fn.sequence 41, 45 #} {# fn.list_set_union macro.MACRO1, macro.MACRO2 #}</pre>	<pre>['40', '41', '42', '43', '44', '45']</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO2={# fn.sequence 41, 45 #} {# fn.list_set_left macro.MACRO1, macro.MACRO2 #}</pre>	<pre>['40']</pre>

Continued on next page

Table 1 -- continued from previous page

Example	Output
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO2={# fn.sequence 41, 45 #} {# fn.list_set_right macro.MACRO1, macro.MACRO2 #}</pre>	<pre>['44', '45']</pre>
<pre>MACRO1={# fn.sequence 40, 43 #} MACRO2={# fn.sequence 41, 45 #} {{ fn.list_set_syndiff macro.MACRO1, macro.MACRO2 }}</pre>	<pre>['40', '44', '45', →']</pre>
<pre>{ "input": { "list": ["AUS", "BHR"] } } {# fn.list_filter_fields data/Countries, 8c0hfle2c0deab00da595101, iso_country_code, country_name, input.list #}</pre>	<pre>["Australia", "Bharain"]</pre>
<pre>{ "input": { "args": [["AAA"], ["BBBB", "CCCC"], ["DD"]] } } {# fn.flatten_list_of_lists input.args #}</pre>	<pre>["AAA", "BBBB", "CCCC", "DD"]</pre>

Continued on next page

Table 1 -- continued from previous page

Example	Output
<pre> { "input": { "args": [["100", "12000", "13000",]] } } {{ fn.modulo_list input.args,1000 }} </pre>	<pre> ["12000", "13000"] </pre>
<pre> { "input": { "dataList" = [{ "timeZoneCode": "10", "timeZoneName": "Africa/Bissau" }, { "timeZoneCode": "100", "timeZoneName": "America/Noronha" }, ] } } {{ fn.get_tvpair_list input.dataList, title:timeZoneName, value:timeZoneCode }} </pre>	<pre> [{ "value": ↪ "10", "title": ↪ "Africa / ↪ Bissau" }, { "value": ↪ "100", "title": ↪ "America / ↪ Noronha" }, ] </pre>

9.3.5. 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.

9.3.6. 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"
]
```

9.3.7. 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 (checkbox is enabled).

Refer to the settings topic called `Additional Role Access Profile Validation`.

9.3.8. Macro 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: <pre>{# fn.evaluate self.x #}</pre>	<pre>['ZAF']</pre>

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: <pre>{# fn.evaluate self.x #}</pre>	Codes with same prefix as FRA: <pre>['DNK', 'HKG', 'FRA', 'DEU', 'IND', 'ITA', 'NLD', 'NZL', 'SAU', 'ESP', 'SWE', 'ZAF', 'CHE', 'TUR']</pre>

9.3.9. 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
<pre> hierarchy: sys.hcs.CS-P.CS-NB. Geologic.EMEA.Paar1 {# fn.get_cucms_associated_via_ndlr ↪#} </pre>	<pre> [10.110.11.131] </pre>

- *fn.get_cucm_bkeys_associated_via_ndl*: Get all CUCM server business keys from ND (Network Device List) on current the hierarchy. A business key is a list of: IP, port, hierarchy.

Example	Output
<pre> hierarchy: sys.hcs.CS-P {# fn.get_cucm_bkeys_associated_via_ ↪ndl #} </pre>	<pre> [["10.110.11.131", "8443", "hcs.CS-P.CS-NB.AAAGlobal"]] </pre>

- *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
<pre>{{ fn.cucm_get_line_details 4025, VS-Corp-NewYork, all,57e709467677f0c9ca956f6f}}</pre>	<pre>{ "is_line_shared": true, "remote_destination_profiles": ["RDP_vdevenr"], "device_profiles": ["UDP_vdevenr"], "phones": ["SEP002155D547F7", "SEP111122223333"] }</pre>

Single cluster examples:

Example	Output
<pre>{{ fn.cucm_get_line_details 4025, VS-Corp-NewYork }}</pre>	<pre>{ "is_line_shared": true, "remote_destination_profiles": ["RDP_vdevenr"], "device_profiles": ["UDP_vdevenr"], "phones": ["SEP002155D547F7", "SEP111122223333"] }</pre>
<pre>{{ fn.cucm_get_line_details 4025, VS-Corp-NewYork, is_line_shared }}</pre>	<pre>true</pre>
<pre>{{ fn.cucm_get_line_details 4025, VS-Corp-NewYork, phones }}</pre>	<pre>SEP002155D547F7 SEP111122223333</pre>

- *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

<pre> {{ fn.get_endpoint_name VG350 SCCP SKIGW1122111111 ↪ANALOG 2 SM-D-48FXS-E-SCCP 0 46 ↪'Analog Phone' }} </pre>	AN112211111142E
<pre> {{ fn.get_endpoint_name VG350 MGCP test1.com ANALOG 2 SM-D-72FXS 0 9 'Cisco MGCP ↪FXS Port' }} </pre>	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

<pre> {{ fn.get_sccp_endpoint_name VG350 SCCP SKIGW1122111111 ANALOG 2 SM-D-48FXS-E-SCCP 0 46 'Analog ↪Phone' }} </pre>	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:

<pre>{{ fn.lines_from_hierarchy_devices sys.hcs.CS-P.OBCust.OBSite1 Phone }}</pre>	<pre>{ "pattern": ["555555", "710087"], "routePartitionName": [None, "Site- ↳CPT1"]} }</pre>
--	--

- *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:

<pre>{{ fn.default_device data/CallManager. ↳pkid }}</pre>	<pre>69cer80r903aa8b565784675</pre>
<pre>{{ fn.default_device data/CallManager } ↳}</pre>	<pre>[{ "pkid": "59ccdc58dcbdffaa51eedbed9 ↳", "model_type": "data/CallManager", "uri": "/api/v0/data/CallManager/ ↳69cer80r903aa8b565784675 }]</pre>

- *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:

<pre>{{ fn.ndl_device_meta fn.hierarchy, ndl.data/CallManager.pkid }}</pre>	<pre>69cer80r903aa8b565784675</pre>
<pre>{{ fn.ndl_device_meta fn.hierarchy }}</pre>	<pre>{ "ndl": { "name": "NDL-GeoLogic-1", "pkid": ↪"59ccdc5303aa8b5657426ac", "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\"]" } } }</pre>

9.3.10. 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”: TCTDOMALLEY218

- 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': TCTFOO00183752

- The generated name is checked against existing device names and the random number is regenerated until the name is unique.

Example	Output
<pre>{{ fn.jabber_device_name 'Cisco Dual Mode for iPhone', D'Malleybazfrobbleverylongname }}</pre>	TCTDOMALLEY218

9.3.11. 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
<pre>{{ fn.process_subscriber_line_data input }}</pre>	<pre>[{ "pattern": "10003", "routePartitionName": "Site-23m-Customer 1 Site A" }, { "pattern": "10005", "routePartitionName": "Site-23m-Customer 1 Site A" }]</pre>

9.3.12. Zero, Unset, Boolean, Drop, Null and Exists Functions

Function	Description	Example	Output or Result
fn.zero	Return a zero value.	<code>{{fn.zero}}</code>	0
fn.unset	Return an empty string.	<code>{{fn.unset}}</code>	' '
fn.true	Return a boolean True.	<code>{{fn.true}}</code>	true
fn.false	Return a boolean False.	<code>{{fn.false}}</code>	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	<code>{{fn.is_none_or_empty ↪input.field1}}</code>	input 1. false input 2. true input 3. true

Given input context:

```
{
  "input": {
    "test": null
  }
}
```

Function	Description	Example	Output or Result
fn.null	Returns Null. Useful in comparison tests.	<code>((input.test == fn.null))</code>	true

Useful in Configuration Templates:

Function	Description	Example	Output or Result
fn.drop	Removes the attribute	<code>{'name': '{{fn.drop}}'}</code>	Attribute dropped
fn.force_null	Attribute has Null value	<code>{'name': '{{fn.force_null}}' ↪'}</code>	<code>{'name': None}</code>

Example of `fn.drop` in an if-then-else test from a Configuration Template of a workflow. the attribute is dropped if not entered as input:

```
"ldapDirectoryName": "( ( input.cucm_user_ldapDirectoryName != '' ) )
<{{ input.cucm_user_ldapDirectoryName }}>
<{{ fn.drop }}"
```

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 exists and has a non-null value.	<code>{{ fn.exists input.field1. ↪key1 }}</code>	input 1. true input 2. false input 3. false
fn.exists	If the <i>argument does not exist</i> , it is interpreted as a string - which is interpretable, so the function returns true.	<code>{{ fn.exists field.key }}</code>	input 1, 2, 3 true

9.3.13. Time Functions

- `fn.now`: Return the date and time at this moment. An optional format parameter is available.

Example:

Example	Output
<code>{{fn.now}}</code>	2013-04-18 10:50:52.105130
<code>{{fn.now "%Y%m%d"}}</code>	20140327
<code>macro.DAY="%A %m/%d/%Y"</code>	
<code>{{fn.now macro.DAY}}</code>	"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 l:M:S p
%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

Continued on next page

Table 2 -- continued from previous page

%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.seconds_to_text*: Given an integer seconds value, convert to days, hours, minutes, seconds

Examples	Output
<code>{{ fn.seconds_to_text 345435 }}</code>	3 days, 23 hours, 57 minutes, 15 seconds
<code>{{ fn.seconds_to_text 8000 }}</code>	2 hours, 13 minutes, 20 seconds
<code>{{ fn.seconds_to_text 35 }}</code>	35 seconds

- *fn.add_x_days_to_startdate*: Given three arguments,
 - integer number of days (positive or negative value)
 - start date
 - date-time format specification

return a date in the past or future.

The arguments can be named macros.

Examples	Output
<pre>{{ fn.add_x_days_to_startdate 10, ↳ '2019-10-01', '%Y:%m:%d:%I:%M' }}</pre>	2019:10:11:02:09
<pre>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 }}</pre>	2019:10:11:02:09

9.3.14. 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.

Examples:

Example	Output
<code>{{fn.hierarchy}}</code>	<code>'52162d552afa433946245bcb'</code>
<code>{{fn.hierarchy_parent}}</code>	<code>'52162d522afa433941245ba0'</code>
<code>{{fn.hierarchy_path}}</code>	<code>['1c0efeg2c0deab10da595101', '52162d4c2afa433940245ba3', '52162d4e2afa43393b245ba2', '52162d522afa433941245ba0', '52162d552afa433946245bcb']</code>
<code>{{fn.hierarchy_parent_path}}</code>	<code>['1c0efeg2c0deab10da595101', '52162d4c2afa433940245ba3', '52162d4e2afa43393b245ba2', '52162d522afa433941245ba0']</code>
<code>{{fn.hierarchy_friendly_path}}</code>	<code>'sys.GenCorp.SuperCom.ABCGroup. Branch1'</code>
<code>{{fn.hierarchy_friendly_parent_path}}</code>	<code>'sys.GenCorp.SuperCom.ABCGroup'</code>
<pre>Hierarchy = sys.hcs.CS-P.CS-NB.AAAGlobal {# fn.friendly_path_choices ,down #}</pre>	<pre>["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"]</pre>
<pre>Hierarchy = sys.hcs.CS-P.CS-NB.AAAGlobal (cust) {{ fn.is_site }}</pre>	<code>false</code>

9.3.15. Request Functions

Function	Description	Example
<code>fn.request_user_name</code>	Return the logged in username.	<code>{{fn.request_user_name}}</code>
<code>fn.request_user_role</code>	Return the logged in user role.	<code>{{fn.request_user_role}}</code>
<code>fn.request_user_email</code>	Return the logged in user email address.	<code>{{fn.request_user_email}}</code>
<code>fn.request_user_pkid</code>	Return the logged in user pkid.	<code>{{fn.request_user_pkid}}</code>

9.3.16. Internal Number Inventory Functions

The `fn.lines` macro functions use 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`, optionally with a specified `routePartitionName`.

All macros 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))`.

- `CUSTOMER_INI_ENABLED` macro is `((False))`:

Give the following patterns and route partition on the Unified Communications Manager:

Number	Partition
1000	
2000	Site-REL103-Customer
3000	
4000	Site-REL103-Customer
5000	Site-REL103-Customer
6000	

- `fn.lines`: Return a list of lines. With no parameter, list all the lines on the associated Unified Communications Manager. If the optional parameter is a route partition, list the lines in the partition. If the optional parameter is `custom`, show an empty list.

If an E164 number is associated with the INI number, then the E164 number is also shown.

Examples:

Example	Output
<code>{{fn.lines}}</code>	<pre>[{u'value': u'1000', u'title': u'1000 +27826543001'} ↔, {u'value': u'2000', u'title': u'2000'}, {u'value': u'3000', u'title': u'3000'}, {u'value': u'4000', u'title': u'4000 +27826543004'} ↔, {u'value': u'5000', u'title': u'5000'}, {u'value': u'6000', u'title': u'6000'}]</pre>
<code>{{fn.lines Site-REL103-Customer}}</code>	<pre>[{u'value': u'2000', u'title': u'2000'}, {u'value': u'4000', u'title': u'4000'}, {u'value': u'5000', u'title': u'5000'}]</pre>
<code>{{fn.lines custom}}</code>	<pre>[{u'value': u'5000', u'title': u'5000'}]</pre>
	<code>[]</code>

- CUSTOMER_INI_ENABLED macro is ((True)):

Given the following properties of an example Internal Number Inventory:

Number	In Use	Available
1000	N	Y
2000	N	Y
3000	N	N
4000	Y	Y
5000	Y	Y
6000	Y	N

- *fn.lines*: Return a list of available lines on the Internal Number Inventory, with used lines indicated as (used). The E164 number associated with the INI is shown if available.
- *fn.lines_available_only*: Return a list of available lines from the Internal Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_used_only*: Return a list of used lines from the Internal Return a Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_unavailable_only*: Return a list of unavailable lines from the Internal Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_unused_only*: Return a list of unused lines from the Internal Number Inventory, with <TITLE> in brackets after each.

- *fn.lines_available_used*: Return a list of available and used lines from the Internal Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_available_unused*: Return a list of available and unused lines from the Internal Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_unavailable_used*: Return a list of unavailable, used lines from the Internal Number Inventory, with <TITLE> in brackets after each.
- *fn.lines_unavailable_unused*: Return a list of unavailable and unused lines from the Number Inventory, with <TITLE> in brackets after each.

Examples:

Example	Output
<code>{{fn.lines}}</code>	<pre>[{u'value': u'1000', u'title': u'1000 +27826543001', {u'value': u'2000', u'title': u'2000'}, {u'value': u'4000', u'title': u'4000 (used) ─ →+27826543004'}, {u'value': u'5000', u'title': u'5000 (used)'}]</pre>
<code>{{fn.lines_available_only avail}}</code>	<pre>[{u'value': u'1000', u'title': u'1000 (avail)'}, {u'value': u'2000', u'title': u'2000 (avail)'}, {u'value': u'4000', u'title': u'4000 (avail)'}, {u'value': u'5000', u'title': u'5000 (avail)'}]</pre>
<code>{{fn.lines_used_only inuse}}</code>	<pre>[{u'value': u'4000', u'title': u'4000 (inuse)'}, {u'value': u'5000', u'title': u'5000 (inuse)'}]</pre>
<code>{{fn.lines_unavailable_only}}</code>	<pre>[{u'value': u'3000', u'title': u'3000'}, {u'value': u'6000', u'title': u'6000 (used)'}]</pre>
<code>{{fn.lines_unused_only}}</code>	<pre>[{u'value': u'1000', u'title': u'1000'}, {u'value': u'2000', u'title': u'2000'}, {u'value': u'3000', u'title': u'3000'}]</pre>
<code>{{fn.lines_available_used}}</code>	<pre>[{u'value': u'4000', u'title': u'4000 (used)'}, {u'value': u'5000', u'title': u'5000 (used)'}]</pre>
<code>{{fn.lines_available_unused}}</code>	<pre>[{u'value': u'1000', u'title': u'1000'}, {u'value': u'2000', u'title': u'2000'}]</pre>
<code>{{fn.lines_unavailable_used}}</code>	<pre>[{u'value': u'6000', u'title': u'6000'}]</pre>

- *fn.get_associated_lines*: Return a list of lines associated with a given userid from device.cucm.User.

Examples:

Example	Output
<pre>{{ fn.get_associated_lines jsmith }}</pre>	<pre>{ "85089", "85090" }</pre>

- *fn.associated_dn_list*: Return a list of directory numbers associated with E164 numbers at the specified context hierarchy and lower. Two models are queried. Results include the range: data/HcsDpDNE164AssociateDAT and data/HcsDpDNMultiE164AssociateDAT.

Examples:

Example	Output
<pre>{# fn.associated_dn_list #}</pre>	<pre>["1015000", "81104600", "81104601", "81104602", "81104603"]</pre>

- *fn.get_dn_number*: Return the matching Directory Number (DN) in a range given a E164 number as input.

Two models are queried. If no results are found in data/HcsDpDNE164AssociateDAT then the ranges in data/HcsDpDNMultiE164AssociateDAT are queried.

Examples:

Example	Output
<pre>hierarchy: sys.hcs.CS.Global.LOC002 {{ fn.get_dn_number \+121000 }}</pre>	<pre>82041000</pre>

- *fn.get_e164_number*: Return the matching E164 number at the specified context hierarchy; given a Directory Number (DN) range as input. If not found, return a blank result.

Two models are queried. If no results are found in data/HcsDpDNE164AssociateDAT then the ranges in data/HcsDpDNMultiE164AssociateDAT are queried. This means that a E164 range mapping to a range of DNs including that DN would take precedence over a range of E164 mapped to a single DN if both mappings existed for the DN. In the event of a match on a range of E164 numbers mapped to a Single DN, the primary E164 configured is the E164 number returned. If the primary is not configured then no E164 number is returned.

Examples:

Example	Output
<pre> hierarchy: sys.hcs.CS.Global.LOC002 {{ fn.get_e164_number 82029712 }} </pre>	<pre> \+1555559712 </pre>

9.3.17. 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-4-UC GUI 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

- *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-4-UC GUI 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 save the phone registration status of these to the VOSS-4-UC database.

Examples:

Example	Output
<pre>{{fn.get_phone_status 5ca2b90bce894e0014d488fb, status}}</pre>	<pre>"Registered"</pre>
<pre>{{fn.get_phone_statuses 5ca2b90bce894e0014d488fb; 3da7c60bd4632f1113a255dc, status; ip_address}}</pre>	<pre>[{ "status": "Registered", "ip_address": "172.29.90.80" }, { "status": "Registered", "ip_address": "172.29.90.11" }]</pre>

9.3.18. 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 administrator GUI and selfservice GUI.
- *fn.list_installed_languages_admin*: List the installed languages in the administrator GUI.
- *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 GUI
- 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
<pre>{{ fn.localize data.LocalizedModelStore. localized_value where_clause }}</pre>	A localized value is returned.
<pre>{# fn.list_installed_languages #}</pre>	<pre>[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title': 'German'}]</pre>
<pre>{# fn.list_installed_languages_admin #}</pre>	<pre>[{'value': 'en-us', 'title': 'English'}]</pre>
<pre>{# fn.list_installed_languages_selfservice #}</pre>	<pre>[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title': 'German'}]</pre>
<pre>{# fn.list_installed_languages_by_role none #}</pre>	<pre>[{'value': 'en-us', 'title': 'English'}, {'value': 'de-de', 'title': 'German'}]</pre>
<pre> {{ fn.localize_choices data.HcsNbnSupportedModelsDAT.operations modelType:data/NormalizedUser }}</pre>	<p>output before localization:</p> <pre>[{'title': _('create'), 'value': 'create'}, {'title': _('delete'), 'value': 'delete'}, {'title': _('update'), 'value': 'update'}]</pre>

9.3.19. 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 GUI.

The macro is typically added to a workflow “Set” step for debugging purposes.

Examples:

Example	Output
<pre data-bbox="232 310 597 506"> {{fn.log info, This is an informational message.}} {{fn.log debug, Debug message.}}</pre>	<p data-bbox="862 268 1013 296">In app.log:</p> <pre data-bbox="862 317 1279 453"> INFO This is an informational message. DEBUG Debug message.</pre>
<p data-bbox="232 541 837 600">A workflow (1PWF) step Set variable “bar” is a message with the value of “name”.</p> <pre data-bbox="232 621 553 648"> {{fn.txn_log pwf.name}}</pre>	<p data-bbox="927 562 1328 621">Step 1 - Set workflow context (1PWF):</p> <pre data-bbox="862 642 1349 1073"> { "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" }</pre>

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.

9.3.20. Object Functions

- *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
<code>{{ fn.object_keys input.object,true }}</code>	<code>["boolean_1","boolean_3"]</code>
<code>{{fn.object_keys input.object,"1"}}</code>	<code>["string_1","string_1_dup"]</code>
<code>{{fn.object_keys input.object,1}}</code>	<code>["integer_1","integer_1_dup"]</code>
<code>{{fn.object_keys input.object}}</code>	<code>["boolean_1","boolean_2", "boolean_3","string_1", "string_1_dup","string_2", "integer_1","integer_1_dup", "integer_2"]</code>

- *fn.object_empty*: Returns and empty object

Example:

Example	Output
<code>{{ fn.object_empty }}</code>	<code>{}</code>

- *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 does not exist, the pair is added.
- If the key exists, the value is updated.

Examples:

Example	Output
<pre>my_object: { "existing_key": "some_value" } function call: {{ fn.object_update "key", "1234", input.my_object }}</pre>	<pre>{ "existing_key": "some_value", "key": "1234" }</pre>
<pre>my_object: { "key": "some_value" } function call: {{ fn.object_update "key", "1234", input.my_object }}</pre>	<pre>{ "key": "1234" }</pre>

9.3.21. 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'.

9.3.22. 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>.

Examples:

Example	Output
<pre>macro: USA_pkid = {{data.Countries.__pkid iso_country_code:USA}} {{fn.pkid_to_bkey macro.USA_pkid}}</pre>	<pre>"[u'United States of America', u'USA', u']"</pre>
<pre>macro: a_country_bkey = {{data.Countries.__bkey __pkid:macro.USA_pkid}} {{fn.bkey_to_pkid macro.a_country_bkey, data/Countries}}</pre>	<pre>"52d3eba8893d57373f842acb"</pre>
<pre>context data: { "self": { "bk": "[\"10.110.21.101\", \"8443\", \"hcs.CS-P.CS-NB.AAAGlobal\"]" } } function: {{ fn.from_business_key_format self.bk }}</pre>	<pre>["10.110.21.101", "8443", "hcs.CS-P.CS-NB.AAAGlobal"]</pre>

Example	Output
<code>{{fn.as_int "1"}}</code>	1
<code>{{fn.as_string 12345}}</code>	"12345"
<code>{{fn.as_bool "T"}}</code> <code>{{fn.as_bool "0"}}</code>	true false
<code>{#fn.as_list foo bar#}</code> <code>{#fn.as_list 'foo,bar'#}</code> <code>{#fn.as_list ['foo','bar']#}</code>	['foo bar'] ['foo,bar'] ['foo','bar']
<code>{{fn.int_to_hex 255}}</code>	ff
<code>{{fn.hex_to_int ff}}</code>	255
<code>{{ fn.getMajorMinorVersion 10.5(4) }}</code> <code>{{ fn.getMajorMinorVersion 11.5(1) SU1 }}</code> <code>{{ fn.getMajorMinorVersion 11.5.3 }}</code>	10.5 11.5 11.5

9.3.23. 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 omitted 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",

```

(continues on next page)

(continued from previous page)

```

    "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, AAAaaaBBBaaaCCc }}
```

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"
  }
]

```

9.3.24. HTTP GET Function

- *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"
}
```

9.3.25. 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'
```

9.3.26. 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',
```

(continues on next page)

(continued from previous page)

```

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 #}

[{'country_name': u'Australia',
  u'iso_country_code': u'AUS'},
 {'country_name': u'Bahrain',
  u'iso_country_code': u'BHR'},
 {'country_name': u'Canada',
  u'iso_country_code': u'CAN'},
 {'country_name': u'Denmark',
  u'iso_country_code': u'DNK'},
 ...
 ...
 {'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'#}

[{'pattern': u'55554444',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'8100240105',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'5544332211',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'55667722',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'8765653',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'66776767',
  u'routePartitionName': u'NullPartition'},
 {'pattern': u'3009',

```

(continues on next page)

(continued from previous page)

```

    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}'}}]},
  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}'},

```

(continues on next page)

(continued from previous page)

```
{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'}}
```

9.4. Macro Examples and Use

9.4.1. Create a Macro In-line

To create a macro in-line, 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.

9.4.2. Create 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.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}}`.

9.4.3. 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 “}}”.

9.4.4. Substitution Macro Examples

```
{{CallManager.host}}
http://{{CallManager.host}}
http://{{CallManager.host}}/{{CallManager.username}}
```

9.4.5. Create 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.

9.4.6. 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”

9.4.7. Macro Evaluator

The Macro Evaluator is available to test a macro at a hierarchy level. Refer to the macro functions, syntax and examples in the documentation.

1. Choose **Administration Tools > Macro Evaluator** to open the Macro Evaluator input form.
2. Enter the macro text in the Macro input box. By default, a macro `{# data.Countries.* #}` is entered as a test macro to evaluate.
3. From the drop-down list, choose the Context Hierarchy at which the macro is to be evaluated.
4. Enable and provide values for Context Data if required. Refer to the examples and Designer Guide for macro reference topics.
5. Click **Evaluate** to run the query.

The result of the evaluated macro is displayed in the Output box.

Index

M

Macro function

- fn.add, 95
- fn.add_x_days_to_startdate, 117
- fn.as_bool, 132
- fn.as_int, 132
- fn.as_list, 100, 132
- fn.as_string, 132
- fn.associated_dn_list, 126
- fn.bkey_to_pkid, 132
- fn.contains, 96
- fn.containsIgnoreCase, 96
- fn.containsStartOf, 97
- fn.containsStartsWith, 97
- fn.cucm_get_line_details, 110
- fn.default_device, 110
- fn.divide, 96
- fn.drop, 116
- fn.evaluate, 109
- fn.exists, 117
- fn.false, 116
- fn.filter_by_rule, 106
- fn.filter_roles_by_user_access_profile, 108
- fn.fix_non_ascii, 98
- fn.fix_username, 99
- fn.flatten_list_of_lists, 100
- fn.force_null, 116
- fn.friendly_path_choices, 120
- fn.from_business_key_format, 132
- fn.get_admin_roles_allowed_at_hn, 107
- fn.get_associated_lines, 126
- fn.get_cucm_bkeys_associated_via_ndl, 110
- fn.get_cucms_associated_via_ndlr, 110
- fn.get_dn_number, 126
- fn.get_e164_number, 126
- fn.get_endpoint_name, 110
- fn.get_phone_status, 10, 127
- fn.get_phone_statuses, 127
- fn.get_sccp_endpoint_name, 110
- fn.get_tvpair_list, 100
- fn.getMajorMinorVersion, 132
- fn.group_by_larger_than_count, 99
- fn.hex_to_int, 132
- fn.hierarchy, 120
- fn.hierarchy_friendly_parent_path, 120
- fn.hierarchy_friendly_path, 120
- fn.hierarchy_parent, 120
- fn.hierarchy_parent_path, 120
- fn.hierarchy_path, 120
- fn.index, 96
- fn.int_to_hex, 132
- fn.is_none_or_empty, 116
- fn.is_site, 120
- fn.isexactly, 97
- fn.jabber_device_name, 114
- fn.join, 96
- fn.length, 96
- fn.lines, 123
- fn.lines_available_only, 123
- fn.lines_available_unused, 124
- fn.lines_available_used, 123
- fn.lines_from_hierarchy_devices, 110
- fn.lines_unavailable_only, 123
- fn.lines_unavailable_unused, 124
- fn.lines_unavailable_used, 124
- fn.lines_unused_only, 123
- fn.lines_used_only, 123
- fn.list_append, 99
- fn.list_contain, 99
- fn.list_count, 99
- fn.list_count_item, 99
- fn.list_empty, 100
- fn.list_extend, 100
- fn.list_extend_no_dup, 100
- fn.list_filter_fields, 100
- fn.list_in, 99
- fn.list_index, 99
- fn.list_index_item, 99
- fn.list_insert, 99
- fn.list_insert_no_dup, 99
- fn.list_installed_languages, 128
- fn.list_installed_languages_admin, 128

-
- fn.list_installed_languages_by_role, 128
 - fn.list_installed_languages_selfservice, 128
 - fn.list_pop, 99
 - fn.list_remove, 100
 - fn.list_remove_dup, 100
 - fn.list_remove_nulls, 100
 - fn.list_reverse, 100
 - fn.list_set_intersect, 100
 - fn.list_set_left, 100
 - fn.list_set_right, 100
 - fn.list_set_syndiff, 100
 - fn.list_set_union, 100
 - fn.list_sort, 100
 - fn.localize, 128
 - fn.localize_choices, 128
 - fn.log, 129
 - fn.lower, 96
 - fn.mask, 96
 - fn.maxval, 95
 - fn.minval, 95
 - fn.modulo_list, 100
 - fn.multiply, 96
 - fn.now, 117
 - fn.null, 116
 - fn.object_empty, 131
 - fn.object_keys, 131
 - fn.object_update, 131
 - fn.one, 100
 - fn.pkid_to_bkey, 132
 - fn.process_subscriber_line_data, 115
 - fn.replace, 97
 - fn.request_get, 136
 - fn.request_user_email, 122
 - fn.request_user_name, 122
 - fn.request_user_pkid, 122
 - fn.request_user_role, 122
 - fn.seconds_to_text, 117
 - fn.sequence, 100
 - fn.split, 96
 - fn.sub_string, 96
 - fn.subtract, 96
 - fn.title, 96
 - fn.true, 116
 - fn.txn_log, 129
 - fn.unset, 116
 - fn.upper, 96
 - fn.validate_name, 97
 - fn.zero, 116
 - fn.zeropad, 95
- Macros (*Feature*)
- Call Pickup Groups Site Defaults, 72
 - Hot Dial PLAR Site Defaults, 72
 - Hunt Groups Site Defaults, 72
 - Lines Site Defaults, 66
 - Named Macros - Site Defaults Line, 81
 - Named Macros - Site Defaults CUC, 83
 - Named Macros - Site Defaults Device, 80
 - Named Macros - Site Defaults User, 82
 - Named Macros Available to Administrators, 79
 - Quick Subscriber Site Defaults, 70
 - Subscriber Site Defaults, 68
 - Voicemail Site Defaults, 71
 - Webex Site Defaults, 71