Quick answers to common problems

Cisco Unified Communications Manager 8: Expert Administration Cookbook

Over 110 advanced recipes to effectively and efficiently configure and manage Cisco Unified Communications Manager

Tanner Ezell

[PACKT] PUBLISHING enterprise
professional expertise distilled

Cisco Unified Communications Manager 8: Expert Administration Cookbook

Over 110 advanced recipes to effectively and efficiently configure and manage Cisco Unified Communications Manager

Tanner Ezell

[PACKT] PUBLISHING enterprise professional expertise distilled

BIRMINGHAM - MUMBAI

Cisco Unified Communications Manager 8: Expert Administration Cookbook

Copyright © 2012 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing and its dealers and distributors, will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: March 2012

Production Reference: 1170312

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-432-3

www.packtpub.com

Cover Image by David Gimenez (bilbaorocker@yahoo.co.uk)

Credits

Author
Tanner Ezell

Reviewers
Michael Ciulei
John Schreiner
Victor Rosa
Justin O'Sullivan

Acquisition Editor
Rukshana Khambatta

Lead Technical Editor
Dayan Hyames

Technical Editors
Veronica Fernandes
Vishal D'souza
Sonali Tharwani

Project Coordinator
Leena Purkait

Proofreader
Lydia May Morris

Indexer
Hemangini Bari

Production Coordinator
Nilesh R. Mohite

Cover Work
Nilesh R. Mohite

About the Author

Tanner Ezell has over five years of experience working with Cisco's Unified Communications platform, along with custom application development in the same field. He currently works as a Consultant with a Cisco partner that specializes in Unified Communications and Contact Center. He spends most of his time designing and implementing UC solutions as well as custom applications and Contact Center scripts. He has a passion for understanding the underlying technology that makes the solutions possible. When he is not working, he enjoys bicycling around the bay area and spending time outdoors on beautiful Northern California days.

I would like to thank my family and my dear friends who have been supportive of this challenging process from the beginning. I cannot thank the contributors, editors, and reviewers enough for the dedication, input, and honest feedback.

I would also like to thank my colleagues, who are a wealth of information, and an uninterruptable source of laughs and good times.

Lastly, I would like to thank my Editor and Project Coordinator who have graciously put up with seemingly endless delays, and offered me this opportunity to grow in ways that will continue to benefit me and my career.

About the Reviewers

Michael Ciulei has more than 11 years of experience in the IT field, as an IT and Telephony Specialist, with a focus on network design and implementation. He is currently a Cisco CCNP Voice Engineer, with a specialization in Advanced Wireless LAN. He has a Bachelor of Engineering degree in Informatics and Computer Science from Tibiscus University, Timisoara.

John Schreiner is a Captain in the United States Marine Corps and the Lead Voice Instructor and Curriculum Manager for the Communication Training Centers. He serves as the Director for Communication Training Center-2, training marines on the East Coast in the latest commercial technologies (Cisco, Microsoft, REDCOM, Harris, and so on). He has experience with both military tactical voice systems and commercial voice systems. He also instructs on security and advanced networking.

John holds CCNP, CCNP Voice, CCNA Security, CCDA, CISSP, and various other certifications.

I'd like to thank my amazing wife, Jacki, who has an impressive tolerance for my Cisco endeavors.

Victor M. Rosa has been in the computer industry for over 25 years. He is experienced in the design and implementation of large IP and multiprotocol networks, encompassing complex wide area network designs, campus local area networks, IP Telephony and Data Center technologies. In the past 10 years, he has planned, implemented, and supported many networks, utilizing both Layer 2 and Layer 3 techniques. In the past five years, he has worked with companies such as HP, Bank of America, and Cisco Systems. He is currently a Networking Consultant for the largest privately owned enterprise in South Carolina, serving the health care industry, pharmaceuticals, and state government agencies. He is certified in two major network technologies: Microsoft and Cisco in the area of Cisco IP Telephony, holding a CCNP Voice and is currently pursuing a CCIE in Routing and Switching.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

PACKTLIB®

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following `@PacktEnterprise` on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface

Cisco Unified Communications Manager 8: Expert Administration Cookbook is filled with many advanced recipes to effectively and efficiently configure and manage Cisco Unified Communications Manager.

This book intends to serve as a quick reference for consultants and administrators to quickly address and resolve common problems while providing design insights. Coupled with clear instructions and plenty of screenshots, this book will help in implementing new features and improving the existing architecture. This practical cookbook will help familiarize the readers with various aspects and conventions of Cisco's Unified Communications Manager solution.

What this book covers

Chapter 1, *Call Routing, Dial Plan, and E.164*, will expose you to call routing with an emphasis around local route groups and E.164. You will learn how to implement least cost routing, Tail End Hop Off, and various call routing technologies available.

Chapter 2, *Call Admission Control*, focuses on call admission control features, the components that make them up, as well as how to reroute calls when enough bandwidth is not available.

Chapter 3, *Media Resources and Music On Hold*, focuses on the multimedia aspects of the platform; you'll learn how to set up Music on Hold and upload custom audio files. You'll also learn how to configure media-related devices and their functions such as Media Termination Points and Transcoders.

Chapter 4, *Tracing and Troubleshooting Tools*, will expose you to some of the most common tools used to debug and troubleshoot issues on the platform, including the Real-Time Monitoring Tool.

Chapter 5, *Device and Unified Mobility*, focuses specifically on mobility for devices and end users. You'll learn to configure single number reach, two state dialing, and how to configure mobility-related features.

Chapter 6, *User Management*, will teach you how to manage end-user permissions, roles and user groups, and how they might apply to end users and administrators alike. You'll learn about LDAP integration and authentication as well as how and when to apply filters.

Chapter 7, *User Features*, focuses on commonly requested and demanded features for users, including Meet-Me conferencing, directed call park, as well as user niceties such as custom ringtones and backgrounds.

Chapter 8, *Advanced Features*, explains advanced features of Unified Communications Manager, specifically focusing around extension mobility, call recording, and monitoring along with the introduction of geolocations and logical partition.

Chapter 9, *Securing Unified Communications*, provides common configuration information for securing a Unified Communications Manager cluster. It also includes configuration for phones and conference resources over SRTP.

Chapter 10, *Serviceability, Upgrades, and Disaster Recovery*, aims to cover configuration of alarms and tracing, along with configuration of the three versions of SNMP. It also covers the backup and restore process for the Unified Communications Manager publisher.

Chapter 11, *Bulk Administration Tool*, introduces the Bulk Administration Tool. We will learn to generate CSV files with and without the `bat.xls` spreadsheet, as well as cover the fields required for some of the most common items that are bulk provisioned including devices, user device profiles, analog gateways, and mobility users.

What you need for this book

- Cisco Unified Communications Manager 8.5
- Cisco IP Communicator 8

Who this book is for

If you are a Cisco Unified Communications Administrator or Engineer looking forward for advanced recipes to perform important administration tasks, then this is the best guide for you. This book assumes familiarity with the basics of Cisco's Unified Communications Manager architecture.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: " The pattern `\+[^1]` will match any E.164 number that does not start with a one."

Any command-line input or output is written as follows:

```
dial-peer voice 200 voip
service CCM
incoming-called number 13400
destination-pattern 13400
session target ipv4:192.168.1.5
codec g711ulaw
dtmf-relay h245-alphanumeric
novad
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: " Add a new route list that will serve as the link to the local route groups (**Call Routing** | **Route/Hunt** | **Route List**)."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title through the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Call Routing, Dial Plan, and E.164

In chapter 1 we dive straight into the dial plan with recipes on E.164 globalization, call routing, and call restrictions. We will cover:

- Implementing local route groups with device pools for E.164 call routing
- Implementing E.164 route patterns and partitions
- Implementing E.164 called and calling party transformations
- Implementing least cost call routing using Tail End Hop Off
- Implementing call restrictions with line blocking patterns and calling search spaces
- Implementing short dial numbers
- Implementing time-of-day call routing
- Implementing Forced Authorization Codes
- Implementing Client Matter Codes

Introduction

In this chapter, we will focus on implementing local route groups, device pools, route patterns, and various other call routing technologies with a specific focus on building an E.164 compatible dial plan. All the recipes in this chapter require administrator access to the **Unified Communications Manager** (**UCM**). It is strongly recommended you get comfortable performing these recipes in a lab environment before implementing them into production.

Even if you're not interested in E.164, the recipes in this chapter can be applied to building any style of dial plan while utilizing some of the feature benefits to make dial plan management easier than before.

Implementing local route groups with device pools for E.164 call routing

To simplify call routing and dial plan management, local route groups provide a logical way to process calls according to settings specific to the device pool of the originating device.

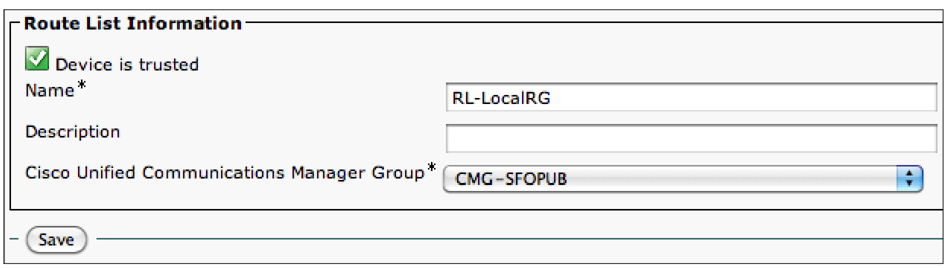
Getting ready

This recipe assumes you have a gateway or trunk device configured.

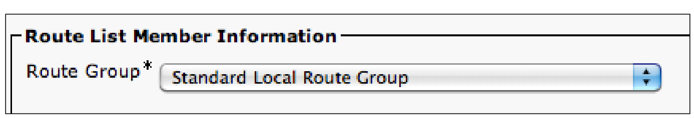
How to do it...

To implement a local route group for use with a device pool, perform the following:

1. Add a new route list that will serve as the link to the local route groups (**Call Routing** | **Route/Hunt** | **Route List**).
2. Click on **Add New** to add a new route list.
3. Type in a name and select a Call Manager Group in the drop-down with which the route list will be associated:

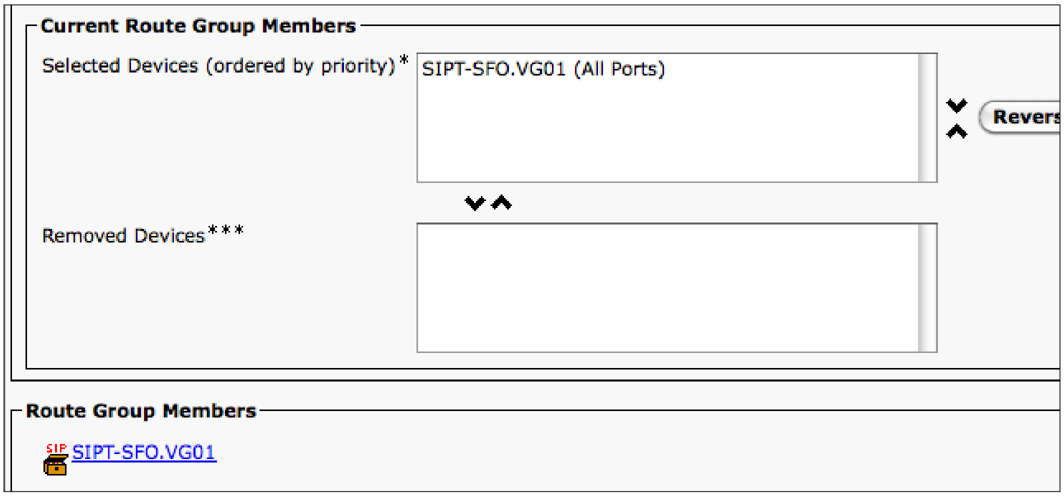


4. Click on **Save**.
5. Once the page reloads, click on **Add Route Group** and a new page will open.
6. Select **Standard Local Route Group** from the **Route Group** drop-down menu then click on **Save**. You will be returned to the **Route List** page:

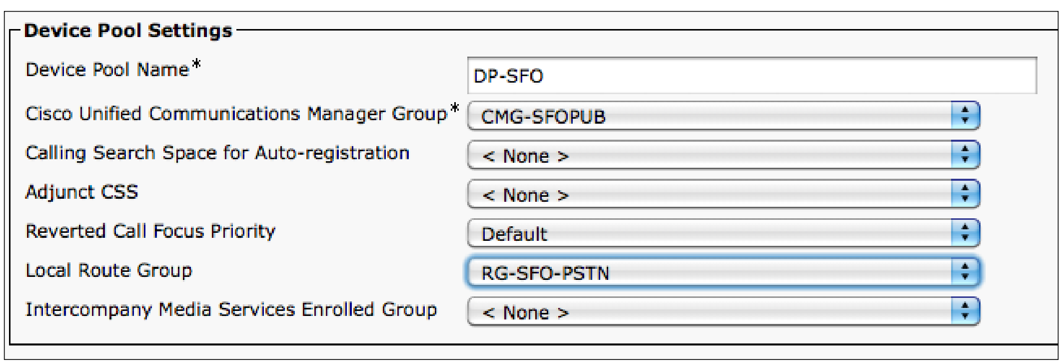


7. Finally, click on **Save** to save the Route List.
8. Add a new route group containing the gateway or trunk (**Call Routing** | **Route/Hunt** | **Route Group**).

9. Find and select your gateway or trunk under the **Find Devices to Add to Route Group** section. Then click on **Add to Route Group**. You should now see the device in the **Selected Devices** list:



10. Click on **Save**. The device will show up under **Route Group Members**.
11. Assign the route group you created in the previous step to the device pool by navigating to the device pool (From the menu, **System** | **Device Pool**) configuration page and selecting the route group from the **Local Route Group** drop-down under the **Device Pool Settings** section:



12. Click on **Save**.

[These changes will not take effect until you *reset* the devices in the device pool.]

How it works...

Prior to the introduction of local route groups in CUCM, dial plans relied on route patterns pointing to specific gateways and route lists in site-specific partitions. By utilizing local route groups with device pools we can simplify call routing and reduce the number of route patterns needed throughout the system, thereby making the overall system simpler and maintenance easier.

There's more...

When a call is placed on the system it matches a route pattern that informs the system where to send the call, typically a route list containing trunks and gateways. When the system is told to send the call to a route list containing the Standard Local Route Group, the egress gateway is determined by information pulled from the device pool settings of the device that initiated the call, and routes it accordingly.

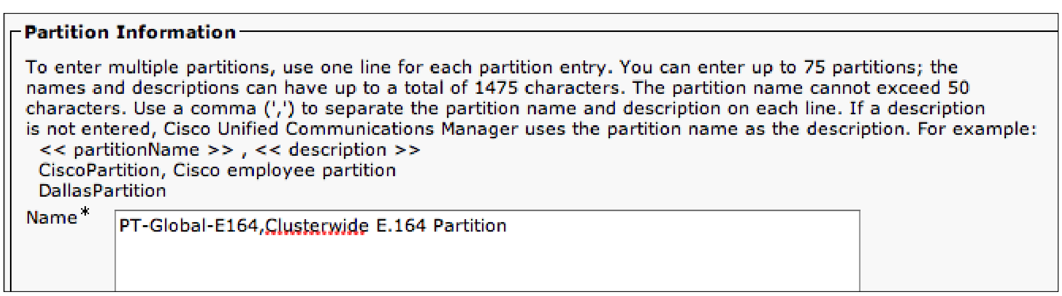
Implementing E.164 route patterns and partitions

An advantage of an E.164 dial plan is that it requires only a single route pattern to make it all work, though additional route patterns are still needed to allow users to dial using traditional dialing and TEHO. Here we will create the route partition to be used by the E.164 route pattern.

How to do it...

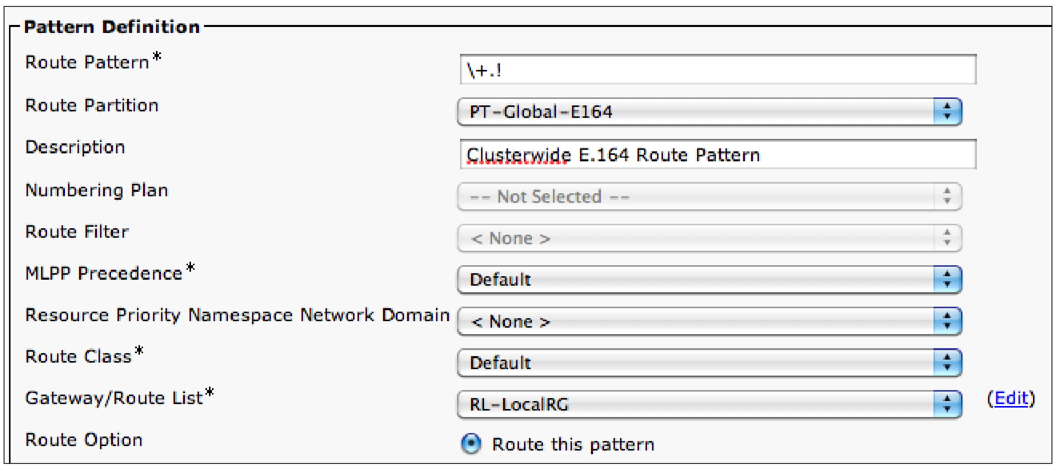
To create the route pattern to support an E.164 dial plan, we will do the following:

1. Add a new partition, which will be globally accessible, by clicking **Add New** on the **Partitions** page located in the **Class of Control** submenu under the **Call Routing** menu.



2. Enter in a partition name and a description in the text box and then click on **Save**.

3. Add the E.164 Route Pattern and assign the Route List to it (**Call Routing** | **Route/ Hunt** | **Route Pattern**).
4. Click on **Add New**.
5. Enter `\+.!` for the **Route Pattern** and select the route partition previously created in the **Route Partition** drop-down:



6. From the **Gateway/Route List*** drop-down, select the route list containing the Standard Local Route Group.
7. Ensure that the **Call Classification** is **OffNet** and the **Route Option** is set to **Route this pattern**.
8. Click on **Save**.

How it works...

When an E.164 number is dialed, the system will match it against the route pattern. The purpose of this pattern is to get the call to route to the local gateway or trunk where number normalization occurs, before sending the call out to the local gateway. **Call Classification** is set to **OffNet** for this pattern because we expect any calls that match this pattern to be routed out to the PSTN.

There's more...

Implementing a successful dial plan requires a few considerations from a technical perspective as well as a user experience standpoint.

Dial plan considerations and partitions

Partitions are a crucial part of both the dial plan and the implementation of calling restrictions. Having a well designed partition scheme can make management easier and it isn't difficult to implement. Some things to consider when planning your partition scheme are as follows:

- How many locations?
- Multinational?
- Will short dials (or hot numbers) be used?
- What about multinational dialing considerations?

Common system partitions

In most systems there are a few basic requirements from a partitioning perspective and at the very least we want to separate user directory numbers from system numbers. To accomplish this we might have the following partitions:

- PT-Line
- PT-System

If this is an E.164 dial plan, we want to separate the partitions from the rest of the system. That is why we also include:

- PT-Global-E164

Partitioning at a national level

In order to support a basic multinational dial plan we need partitions for dialing rules specific to each nation, for example:

- PT-US-DialPlan
- PT-UK-DialPlan

We would typically use these partitions for any patterns that reach the PSTN, including emergency and information services, as well as regular outbound calls.

Partitioning at a local level

If location specific dial rules are required, we might have partitions for each location. For example:

- PT-US-SFO-DialPlan

By doing this at the location level, we can allow for location specific short dials or dialing rules. For example, if we wanted to implement extension 4357 as a short dial to reach the local help desk, we would use a location specific partition such as that shown previously.

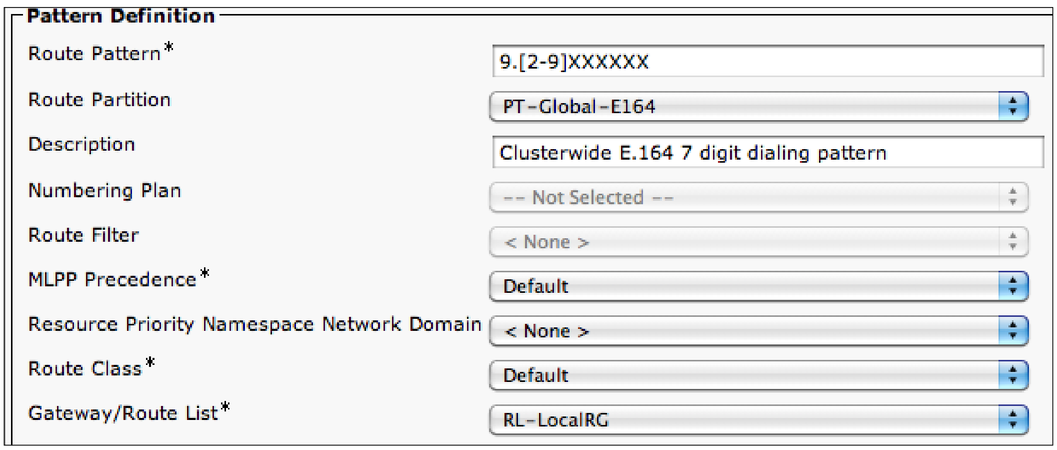
Dial plan considerations and route patterns

It's important to define how users will access the outside world based on what they are familiar with. In many corporations, dial plan rules exist to allow local calls to be dialed first with a 9 or 91, followed by seven or ten digits; other companies may require nine or ten digits for all calls. We call this seven digit and ten digit dialing, respectively.

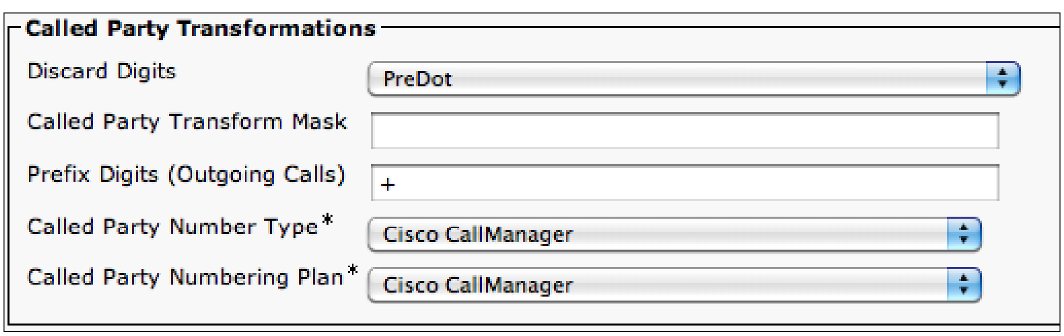
Regardless of which dialing method is used, the setup is the same and thanks to E.164 you only need one route pattern to support all locations.

Seven digit dialing

To implement seven digit dialing we will add another route pattern as explained earlier, which is the `9.[2-9]XXXXXX` pattern:



Unlike the earlier example, we want to strip the `9` off and append a plus sign. This is necessary so the call will match the `\+.!` pattern before it can be routed to the local gateway or trunk:



In situations where you are not using an E.164 dial plan but want to implement seven digit dialing, you need to only put the pattern in a location specific partition and point the **Gateway/Route List*** to the appropriate route list or gateway. In this situation, you would not prefix the plus sign.

> Don't forget to include a pattern for non-local calls!

Ten digit dialing

To configure ten digit dialing, follow the previous steps and instead use a pattern of `9.[2-9]XXXXXXXXX`.

Implementing E.164 called and calling party transformations

By using cluster-wide E.164 route patterns, number localization is no longer done on the route pattern. Instead, localization must occur prior to sending the call to the gateway or trunk device. This is accomplished with called and calling party transformations.

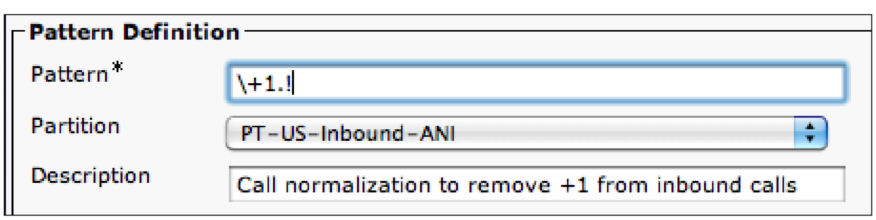
Getting ready

In this recipe we assume you already have the necessary partitions and calling search spaces for called and calling party transformations created. Refer to the *There's more...* section for an example partition and calling search space scheme.

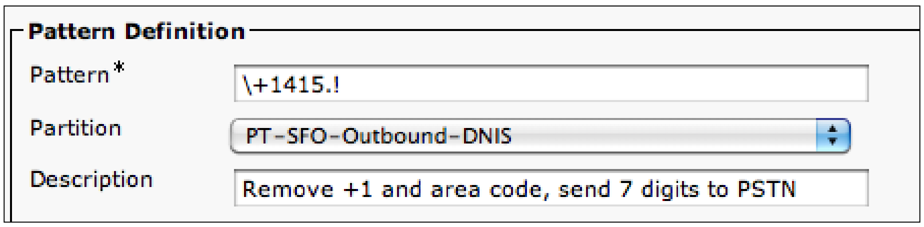
How to do it...

To implement called and calling party transformation on either a gateway or trunk device, perform the following:

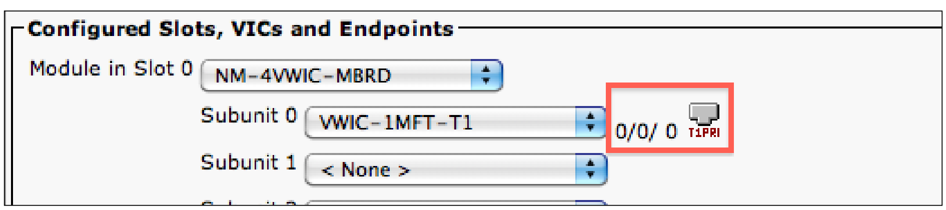
1. Add the calling party transformation pattern (**Call Routing** | **Transformation** | **Transformation Pattern** | **Calling Party Transformation Pattern**).
2. Add the transformation pattern appropriate to your environment and location:



3. Prefix any necessary digits and select the appropriate digit discard field. In the case of the previous example, **Discard Digits** is set to **PreDot** with no digits being prefixed.
4. Add the called party transformation pattern (**Call Routing** | **Transformation** | **Transformation Pattern** | **Called Party Transformation Pattern**).
5. Add the appropriate transformation pattern and any prefix digits necessary. In this case, we again choose **PreDot** for **Discard Digits** and set **Prefix Digits** to **9**. Refer to the *There's more...* section for further explanation if required.



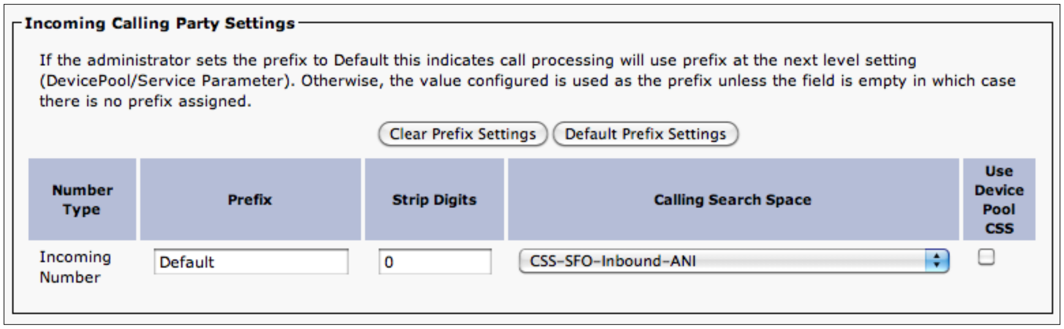
6. Navigate to the configuration page for the port or device.
7. On a MGCP controlled gateway, transformations are configured on a per port basis. The configuration page for the port is found by navigating to the configuration page for the gateway, then selecting the appropriate T1 port under the **Configured Slots, VICs and Endpoints** section as indicated in the following screenshot:



8. This is configured at the device level for trunks and gateways.
9. Next we apply the transformations to our trunks or gateways. Calling party transformations are configured under the section titled **Incoming Calling Party Settings**.

[The type of device we are configuring will determine the fields available to us. On gateway devices we see **National**, **International**, **Unknown**, and **Subscriber**. On trunk devices we see **Incoming Number**.]

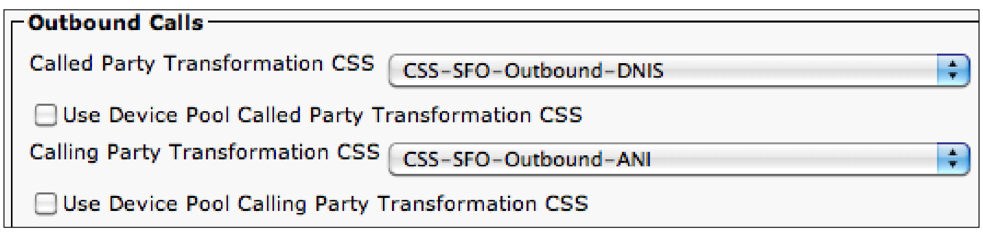
10. Select the **Calling Search Space** that contains the partitions you assigned to the called and calling party transformation patterns and apply it to all applicable fields:



The previous screenshot is for an SIP trunk. Here we uncheck **Use Device Pool CSS** as we are not using the device pool for number transformation.

11. Finally, called party transformations are configured under different sections depending on the type of device.

    On the gateways configuration page, this section is called **Call Routing Information - Outbound Calls**, and **Outbound Calls** for trunks.



Again, we are not using the device pool for number transformation, so we uncheck boxes for both calling and called device pool transformations.

> The calling search space selected for the **Called** and **Calling Party Transformation CSS** must contain the partitions used when creating the transformation patterns.

How it works...

When a call enters through a gateway or trunk device, the calling and called party transformations are applied depending on transformation patterns available to the calling search space used:

- **Calling party transformation patterns**: In the example from the recipe you see a calling party transformation pattern using `\+1.!`. As we explain in the example, we discard digits PreDot. We do this to normalize the number users see when their phone is ringing and connected, as users in the United States may not be accustomed to seeing +1 before the number.

  Alternatively, say we have an office in San Francisco where users are accustomed to seeing only seven digits for local calls and ten for everything else. We still use the `\+1.!` PreDot pattern to remove the +1 for calls but we add another pattern to strip the area code off. In this case that pattern would be `\+1415.!`, stripping PreDot with a partition of PT-SFO-Inbound-ANI, or something similar. By doing this, calls from 415 numbers will show as seven digits on the display when ringing and connected.

- **Called party transformation patterns**: Prior to local route groups and called party transformations, you would prepare the dialed number to be sent to the gateway or route list on the route pattern itself. Called party transformation patterns would then be used to prepare the dialed digits to be accepted by the gateway, trunk, or PSTN. In many cases this involves stripping the plus sign and prefixing an access code before sending it out to the gateway or trunk to route the call to the PSTN.

  How we modify the number depends on the type of gateways or trunks we are using. With MGCP gateways, we format the number so that it can be sent across to the PSTN. In some cases this means removing the plus, and appending or removing digits depending on what the carrier expects. For instance, if the carrier expects seven digits for local calls and 1 + 10 digits for long distance calls, we might strip the +1 and area code for local calls and strip only the plus for all other calls.

  For gateways and trunks, access codes are typically configured to inform the gateway or trunk to send the call to the PSTN. Typically these are 9, or 91. In this situation we would strip the necessary digits and prefix the access code appropriate to the call. For example, say our carrier requires seven digits for local and eleven digits for long distance calls; assuming we require an access code of 9 for local and 91 for long distance calls, we might implement the following called party transformations:

  - \+1.!

    Partition: PT-SFO-Outbound-DNIS

    Prefix digits: 91

  - \+1415.!

Partition: PT-SFO-Outbound-DNIS

Prefix digits: 9

Now, when a call is made to +1 415 555 1234, for example, the transformation pattern will remove +1415 and append a 9, sending the call to the gateway or trunk as 95551234 where it would match a dial peer before being sent out to the PSTN. While it is possible to do these transformations on the gateway themselves, managing them in UCM provides a central point for configuration and can help reduce dial plan maintenance.

There's more...

Calling and called party transformations are primarily used to localize the ANI displayed for calls entering the system and localizing calls for the gateway before sending it out to the PSTN.

Partitions and calling search spaces for called and calling transformation patterns

In this recipe you will see a few partitions and calling search spaces that may not immediately make sense. In order to accomplish these transformations on a per location basis we have six partitions and three calling search spaces.

The partitions and calling search spaces we use are:

- CSS-SFO-Outbound-ANI
  - PT-SFO-Outbound-ANI
  - PT-US-Outbound-ANI
- CSS-SFO-Outbound-DNIS
  - PT-SFO-Outbound-DNIS
  - PT-US-Outbound-DNIS
- CSS-SFO-Inbound-ANI
  - PT-SFO-Inbound-ANI
  - PT-US-Inbound-ANI

If you have no need to localize the ANI on a per location basis you might have a single calling search space and partition instead:

- CSS-US-Inbound-ANI
    - PT-US-Inbound-ANI

These are only some suggestions; make sure you apply the appropriate calling search spaces and partitions for your cluster.

Implementing least cost call routing using Tail End Hop Off

Least Cost Routing (**LCR**) is not strictly limited to calls destined for the PSTN, instead LCR can be used to prevent OnNet calls from being routed OffNet. In this recipe we will cover both uses.

Getting ready

Before we begin this recipe it is helpful to have some information:

- DID ranges of locations for which we are implementing LCR
- Site codes of locations for which we are implementing LCR
- Local numbers per location for Tail End Hop Off

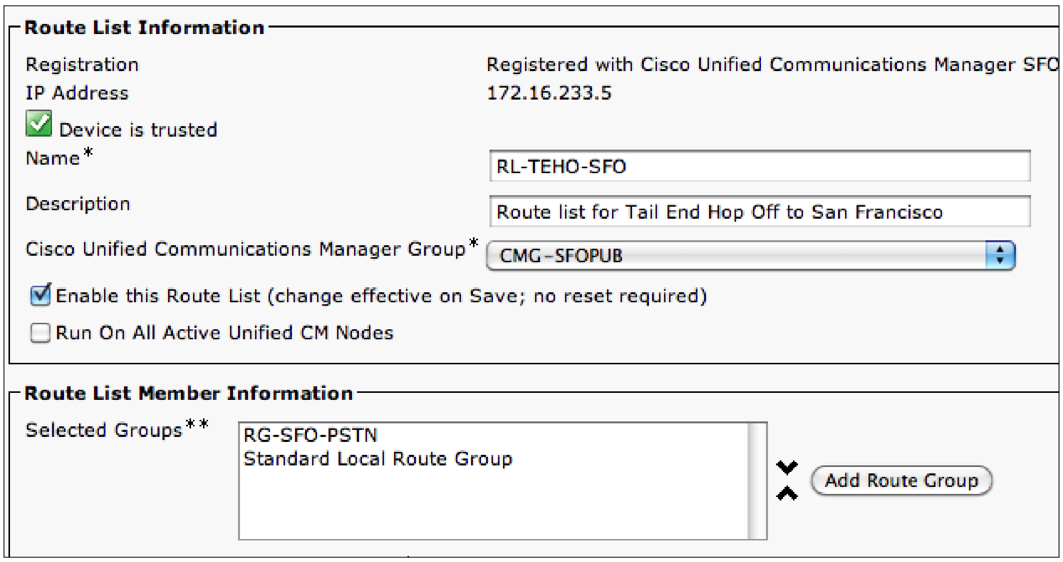
In this recipe we will implement LCR and Tail End Hop Off for calls destined to an office in San Francisco. We will assume the following:

- DID Range for this location: +1 415 555 1000 to 1099
- Site code for this location: 11
- Local numbers for this location: 415 XXX XXXX

How to do it...

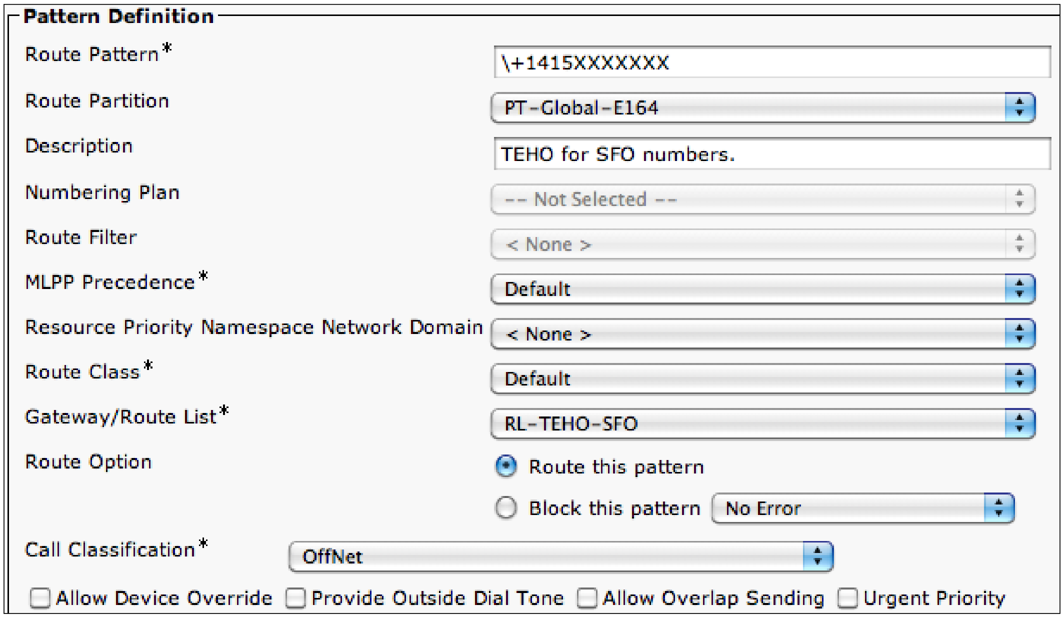
To implement Least Cost Routing for a location, we need to perform the following:

1. Add a new route list that will contain the route group with the gateway or trunk device local to the location for which we are implementing LCR, as well as the Standard Local Route Group:



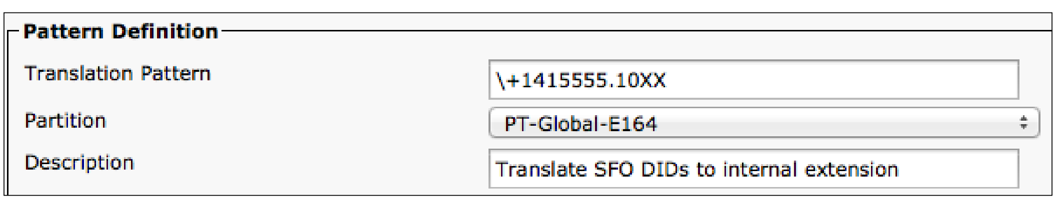
[The order here is important. Ensure the local route group is above the Standard Local Route Group in the list.]

2. Add a new route pattern to send local calls to our new route list. Key fields to note here are **Route Pattern**, **Route Partition**, and **Gateway/Route List***:

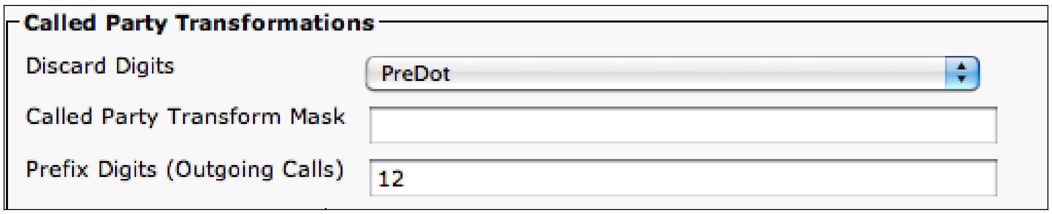


[Here we have unchecked **Provide Outside Dial Tone** as it is unused, but feel free to leave it checked.]

3. Next add a translation pattern (**Call Routing** | **Translation Pattern**, then click on **Add New**) that is responsible for converting E.164 numbers to their internal extensions.
    - Here the **Translation Pattern** must match only the DID range for the location. For our recipe the pattern is `\+1415555.10XX`. For the partition use something that is globally accessible, for example **PT-Global-E164**:



For our pattern, it is necessary to set **Discard Digits** to **PreDot** and **Prefix Digits** to the site code—12 in this recipe.



How it works...

Least cost routing with Tail End Hop off is accomplished by sending calls to locations where the call would cost the least. In addition to Tail End Hop Off, we can accomplish least cost routing by recognizing when a user dials the DID to another user on the same cluster by converting the E.164 number to the local extension and routing over the IP network.

There's more...

Once more we see the benefits of the logical nature of local route groups. By having localization settings at the gateway level, we don't have to worry about formatting and allow the local gateway to normalize the call as required by the PSTN. In the event that the call cannot be made through the gateway or trunk device at the local site, the call will fall back to the gateway or trunk device local to the originating caller.

Do remember that Tail End Hop Off is not legal in all countries. Check with local regulations before implementing it.

Implementing calling restrictions with line blocking partitions and calling search spaces

In this recipe we will be implementing class of service calling restrictions using partitions and calling search spaces, as well as exploring their design considerations.

Getting ready

For this recipe, preparation is key. We will need to determine the partitions, calling search spaces, and patterns to be blocked that will be appropriate to the environment. There is more information on this in the *There's more...* section of this recipe.