



# Cisco Systems Advanced Services

## Cisco CallManager Loop Avoidance—Guide to Best Practices

### Introduction

All telephone systems need to provide methods to prevent calls from infinitely looping within the system and between connected systems. Looping calls consume network and processor resources. In voice over IP (VoIP) networks, this problem is exacerbated. In a pure circuit-switched environment, routing loops eventually consume all available channels and the call begins to clear, but when the call signaling protocol is traveling over a virtual connection, calls can loop indefinitely.

Loops fall into one of the following two categories:

- **Simple routing loops** result when one network node directs calls to adjacent nodes when no end device can be found locally to terminate the call. If similarly configured adjacent nodes end up directing the call back to a previously-visited node, an infinite call loop can result.
- **Forwarding routing loops** occur when a network node can find an endpoint to terminate the call, but a feature on the endpoint diverts the call to another destination. If the diverted-to destination also has diversion configured, the call can forward back (directly or indirectly) to the original endpoint.

### Loop Avoidance Mechanisms in Cisco CallManager

Like most telephone systems, Cisco® CallManager provides several ways to prevent calls from infinitely looping. Table 1 and Table 2 summarize these methods for simple and forwarding loop avoidance.

Table 1 Mechanisms for Simple Loop Avoidance

Mechanism	Description
Calling search spaces and partitions	Configuring inbound trunks to avoid tandem calls can prevent system-to-system routing and end loops as they begin to form.
H.323 pending call account	H.323 calls that receive no significant backward messages are tallied. If the count exceeds a given threshold, the H.323 device refuses to accept more calls and begins clearing calls in the backward direction.



Table 2 Mechanisms for Forwarding Loop Avoidance

Mechanism	Description
ForwardHopCount service parameter	This service parameter controls the maximum number of times a given call can forward <i>within</i> the cluster.
MaxForwardsToDn service parameter	This service parameter controls the maximum number of resolved forward chains that can be pending on a particular number. It helps control forwarding loops <i>between</i> systems that talk over protocols that do not permit the encoding of a forward counter.
QSIGMaximumDiversionCounter service parameter	This service parameter uses QSIG's ability to pass a forward counter from network to network to accurately limit the maximum number of times a call can forward <i>within</i> or <i>between</i> systems.

### Simple Loop Avoidance: Call Search Spaces and Partitions

A five-digit address space permits the configuration of 100,000 distinct addresses. If an administrator had to configure all of these addresses individually, if he added one address per minute, it would take him 16,000 hours to complete the data entry for just a single system. Networks containing multiple systems would complete this data entry for each distinct system in the network.

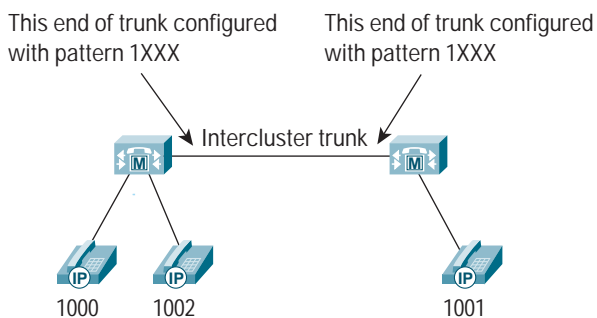
To prevent the administrator from having to enter addresses one by one, telephone systems tend to provide ways to provision a swath of numbers at once. Cisco CallManager relies on regular-expression-based pattern notation to route whole ranges of numbers. If no better match for a digit string is found locally, then Cisco CallManager can route calls to a trunk that is assigned a more general pattern.

This behavior provides customers a good way to distribute their users across systems without having to worry about grouping users rigorously in blocks of numbers. However, this behavior can inflict simple routing loops upon the unwary.

### Simple Loop Avoidance: Dual-System Route Plan Configuration

Figure 1 displays a configuration in which a simple routing loop can occur.

Figure 1  
Dual-System Routing Loop



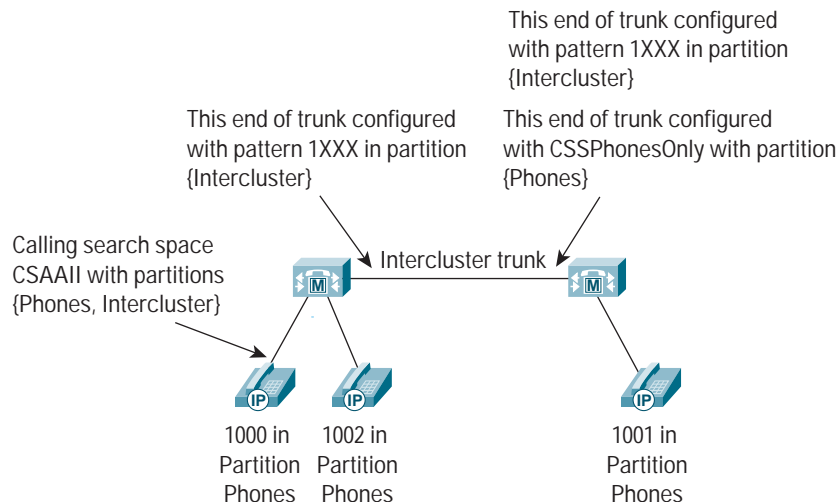


In this figure, if the user at phone 1000 dials 1002, the call routes within the same Cisco CallManager to phone 1002. If, on the other hand, the user at phone 1000 dials 1001, Cisco CallManager can find no local match for 1001 and, instead, matches the pattern 1XXX. This match causes the call to route out the intercluster trunk to the other Cisco CallManager, which extends the call to phone 1001.

However, a problem occurs when a number such as 1003 is dialed. The first Cisco CallManager routes the call over the intercluster trunk to the second Cisco CallManager as normal. However, the second Cisco CallManager is unable to find a match, and the 1XXX pattern associated with the intercluster trunk on the second Cisco CallManager returns the call to the original Cisco CallManager, resulting in an infinite routing loop.

Figure 2 displays how this infinite routing loop can be prevented.

Figure 2  
Calling Search Spaces and Partitions



In this figure, calling search spaces and partitions prevent the routing loop. First, phones and intercluster trunks are isolated within separate partitions: phones within partition phones, intercluster trunks within partition Intercluster.

Next, calling search spaces are assigned in such a way as to prevent looping. Calling search spaces dictate which destinations a caller can dial. A caller can actually call only those endpoints whose patterns or directory numbers reside in a partition that is contained within the caller's search space. Note that, in the example in which 1000 dials 1003, there are really *two* callers. The *first* caller is the phone itself. The second caller is the intercluster trunk at the point at which it enters the second Cisco CallManager, because the first Cisco CallManager ships digits to the second Cisco CallManager via the intercluster trunk, and the intercluster trunk provides these digits to the second Cisco CallManager for analysis.

Because there are two callers in this scenario, there are two calling search spaces: one owned by the calling phone, and one owned by the originating intercluster trunk. Properly setting the calling search spaces prevents the calls from looping.

The calling phone needs access to both the phones on the same Cisco CallManager and the intercluster trunk so that it can reach the second Cisco CallManager if no phone is found locally. Therefore, the calling search space associated with the phone contains both partitions Phones and partition Intercluster.



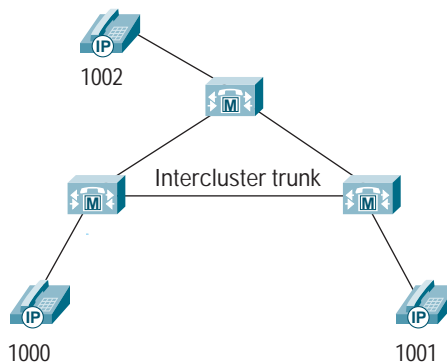
The intercluster trunk, however, needs to avoid calling itself, because a call that comes in an intercluster trunk and immediately routes back out the same intercluster trunk is already half of a loop. The intercluster trunk still needs to be able to reach phones. Therefore, the intercluster trunk's calling search space must not contain its own partition. The intercluster trunk's calling search space, instead, includes only partition Phones. This configuration allows the intercluster trunk to reach the phones on the second Cisco CallManager but not itself, thus stopping routing loops early.

Note that the configuration should be symmetrical. To stop simple routing loops when a phone on the second Cisco CallManager calls the first Cisco CallManager, the intercluster trunk on the first Cisco CallManager should have search space CSSPhonesOnly as well.

### Simple Loop Avoidance: Multiple System Route Plan Configuration

The previous configuration works well for a dual-system configuration, but when more clusters are needed, indirect routing loops can form. Figure 3 displays such a routing loop.

Figure 3  
Multiple System Routing Loop



In this figure, when 1000 dials a number, the number may reside either locally or on two other nodes. From a configuration standpoint, the temptation is to route unknown calls to one of the other two nodes (say, the node controlling 1001) and, if no local match is found there, route the call to the third node. The problem is that if the third node is configured to route unknown calls to the first node, the call loops forever when a non-provisioned number is dialed.

Using the configuration mentioned in the previous section can prevent the loop, in that isolating intercluster trunks within their own partitions can prevent tandem calls. However, using this configuration to prevent tandem calls means that it is not clear how to get a call to the third Cisco CallManager if the second Cisco CallManager receives a call for which there is no local destination.

The solution is either to use gatekeeper-assisted routing (a configuration in which a gatekeeper's routing database determines which Cisco CallManager controls a particular destination) or to configure a hunt list on the first Cisco CallManager that contains the intercluster trunks to both the other Cisco CallManagers. If the Cisco CallManager associated with the first intercluster trunk is unable to find a local destination, it rejects the call. Then, the automated hunting mechanism of the hunt list kicks in, and the first Cisco CallManager extends the call to the remaining Cisco CallManager.



### Simple Loop Avoidance: H.323 Pending Call Count

In the field, one cannot always rely on a loop-safe configuration. In a traditional private branch exchange (PBX) system, loops ultimately self-terminate, because each call reserves one of a finite number of circuits. Eventually, as the call continues to loop, all circuits are consumed and the call terminates.

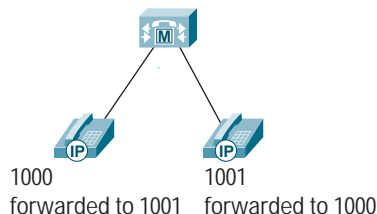
In voice over IP, however, no circuits actually are consumed on a phone-to-phone call. Rather, the signaling exchange permits the endpoints to exchange IP addresses for a media session. As a result, if a signaling loop forms, the call may never clear, because no physical resources—other than memory and CPU, of course—are being consumed as a result of the signaling exchange. When this happens in a Cisco CallManager configuration, CPU ends up spiking to 100 percent indefinitely. Ultimately, the originator detects a problem and attempts to clear the call. However, the situation is like a dog chasing its own tail: the call origination attempt is so far ahead in the loop of the call clearing attempt that the call clearing message can not catch up and terminate the call.

Intercluster trunks, therefore, have a safeguard mechanism. If an intercluster trunk has more than 144 “pending” call originations when a new one arrives, it refuses to process it and, instead, clears the call in the backwards direction. A pending call is one that has received no destination-side message such as ALERTING (which indicates that an actual destination is, in fact, ringing).

### Forwarding Loop Avoidance: MaxForwardHopCount Service Parameter

The previous sections have addressed simple routing loops, but routing loops can occur through another means: call forwarding. Figure 4 displays a typical call forwarding routing loop.

Figure 4  
Typical Call Forwarding Routing Loop



In this figure, if a call arrives for 1000, call forwarding logic redirects the call to 1001. But 1001 is forwarded to 1000, so call forwarding redirects the call back to 1000, and the call loops infinitely.

The service parameter MaxForwardHopCount imposes an upper limit on the number of times that call forward diverts a call. Each time a call forwards to a destination *within the cluster*, the call forward feature increments a counter. When this counter is exceeded, call forward refuses to divert the call any more and either permits the call to ring on the last destination (in the case of call forward no answer, if the destination is actually available) or return busy to the caller (if the destination is not available).

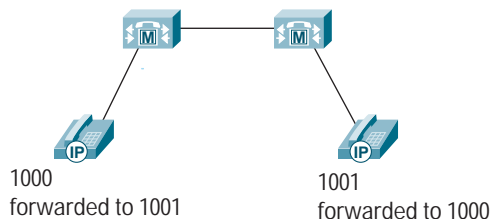
Note that the forwarding logic is not performing loop analysis. The service parameter simply controls the number of diversions, so if one destination is forwarded to a second and is forwarded to a third and so on, at some point, the diversion counter is exceeded and forwarding no longer diverts the call.



### Forwarding Loop Avoidance: MaxForwardsToDN Service Parameter

This setting controls forwarding loops between systems. When a call is forwarded across a gateway or intercluster trunk and later returns, in most protocols Cisco CallManager has no way to determine whether the returning call is one that has been forwarded or one that is a fresh call into the system. Therefore, the MaxForwardHopCount service parameter cannot control the loop, as the incoming call picks up its own forwarding counter. Figure 5 displays a forwarding loop between systems.

Figure 5  
Call Forwarding Loop Between Systems



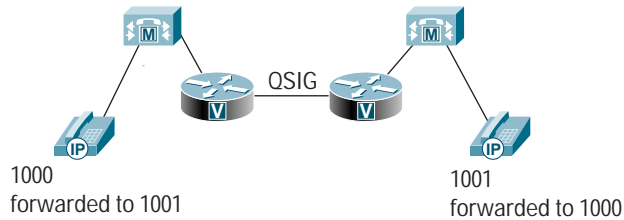
In this case, when a call arrives for 1000, it forwards across the intercluster trunk to the second Cisco CallManager. Forwarding has been configured on 1001, so it forwards the call back across the intercluster trunk. Note that the routing configuration used to control simple route loops does not apply here, because the rerouting across the intercluster trunk is occurring on behalf of 1000's forwarding configuration, not the basic call routing configuration of the intercluster trunk. As the call routes back and forth from Cisco CallManager to Cisco CallManager, new instances of call forwarding objects are instantiated. In each case, the diversion counter for each object is 1, because the forwarding instances seem unrelated. The service parameter MaxForwardsToDN allows the administrator to control these types of loops. This parameter specifies the maximum number of "pending" forwards that can reside on a particular directory number before Cisco CallManager refuses to divert the call. A pending forward is one for which no destination has answered. In the example, while the MaxForwardHopCount for each forward instance is 1, the forwarding logic in the first Cisco CallManager is able to detect that 1000 has N pending forwards, while the forwarding logic in the second Cisco CallManager is able to detect that 1001 has N pending forwards. At some point, the service parameter kicks in and stops the looping. Note that because call forwarding cannot distinguish a call forwarded across systems from a new call freshly arriving, if a particular phone receives a burst of independent calls in excess of the MaxForwardsToDN service parameter, all excess calls do not receive forwarding treatment.

### Forwarding Loop Avoidance: QSIGMaxDiversionCount Service Parameter

QSIG is a protocol designed to support feature transparency between phone systems. As a result, it provides excellent loop avoidance methods. The main problem with Cisco CallManager's detecting forwarding loops is that most protocols do not permit the communication of the diversion count from system to system. QSIG does. Figure 6 displays a call forwarding routing loop in a QSIG environment.



Figure 6  
Call Forwarding Loop in QSIG



In this figure, Cisco CallManagers are connected by gateways running the QSIG protocol. (A more typical configuration would have a PBX on one end of the connection.) As the call forwards from system to system, the QSIG protocol permits each system to encode its current diversion counter. Given this information, the subsequent system can use the encoded information as a seed for its count and thus accurately control the maximum number of times a QSIG call is diverted.

The Cisco CallManager service parameter `QSIGMaxDiversionCount` allows the administrator to define the threshold for forwarding. If this threshold is exceeded, Cisco CallManager ceases diverting the call.

Like `MaxForwardHopCount`, this parameter does not detect loops. Rather, it dictates the maximum number of times a call can be diverted. If a chain of forwarding destinations is configured that contains more links than the value of the service parameter, the call ceases to forward. In addition, if a call forwards across a non-QSIG link, the diversion counter is reset.

### Supported Loop Avoidance Mechanisms by Release

Table 3 lists the different types of loop avoidance mechanisms against Cisco CallManager release.

Table 3 Supported Loop-avoidance Mechanisms by Release

Method	First release
Dual-system configuration	3.0(1)
Multi-system configuration	3.0(1)
H.323 pending call count	3.2(1)
MaxForwardHopCount	3.0(1)
MaxForwardToDNs	3.0(1)
QSIGMaxDiversionCount	3.3(3 FP 2), 4.0(1)

This set of methods provides a robust set of loop avoidance mechanisms. The following section describes one that may be added at a later date.

### Unsupported Loop Avoidance Mechanism: QSIG Transit Call Counter

In addition to stopping forwarding routing loops, QSIG provides a method by which simple routing loops can be terminated. The QSIG transit call counter allows QSIG systems to communicate a hop count from system to system as a call is being routed. If the current hop count exceeds a threshold configured on a system that is processing the call, the system can terminate the call.

Note that this mechanism does not detect loops but rather limits the maximum number of nodes that a call can tandem through. If two endpoints are separated by a long, non-looping chain of systems, a call may not reach its destination. In addition, if the call ever traverses a non-QSIG link, the current hop counter is reset.



**Corporate Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
www.cisco.com  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 526-4100

**European Headquarters**

Cisco Systems International BV  
Haarlerbergpark  
Haarlerbergweg 13-19  
1101 CH Amsterdam  
The Netherlands  
www-europe.cisco.com  
Tel: 31 0 20 357 1000  
Fax: 31 0 20 357 1100

**Americas Headquarters**

Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
www.cisco.com  
Tel: 408 526-7660  
Fax: 408 527-0883

**Asia Pacific Headquarters**

Cisco Systems, Inc.  
Capital Tower  
168 Robinson Road  
#22-01 to #29-01  
Singapore 068912  
www.cisco.com  
Tel: +65 6317 7777  
Fax: +65 6317 7799

**Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the Cisco Web site at [www.cisco.com/go/offices](http://www.cisco.com/go/offices)**

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

All contents are Copyright © 1992–2004 Cisco Systems, Inc. All rights reserved. Cisco, Cisco Systems, the Cisco Systems logo, and Cisco IOS are registered trademarks or trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0304R) ETMG 203246—CM 01.04