Cisco TelePresence External Policy on Cisco VCS

Deployment Guide

Cisco VCS X8.1

D14854.04

December 2013

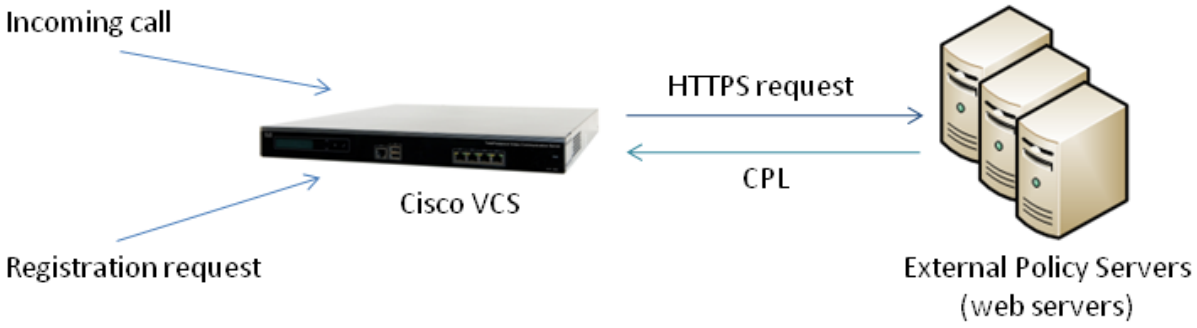
Contents

External policy overview

The Cisco TelePresence Video Communication Server (VCS) has built in support for Registration Policy, Call Policy and User Policy (FindMe) configuration. It also supports CPL (Call Processing Language) for implementing more complex policy decisions. CPL is designed as a machine-generated language and is not immediately intuitive; while the VCS can be loaded with CPL to implement advanced call policy decisions, complex CPL is difficult to write and maintain.

The VCS's external policy feature allows policy decisions to be taken by an external system which can then instruct the VCS on the course of action to take (such as whether to accept a registration, fork a call and so on). Call policy can now be managed independently of the VCS, and can implement features that are unavailable on the VCS. The external policy server can make routing decisions based on data available from any source that the policy server has access to, allowing companies to make routing decisions based on their specific requirements.

When the VCS is configured to use an external policy server the VCS sends the external policy server a service request (over HTTP or HTTPS), the service will send a response back containing a CPL snippet which the VCS will then execute.



Using an external policy server

The main areas where the VCS can be configured to use an external policy server are:

- Registration Policy – to allow or reject registrations.
- Call Policy (also known as Admin Policy) – to control the allowing, rejecting, routing (with fallback if calls fail) and forking of calls.
- Search rules (policy can be applied for specific dial plan search rules).

Each of these areas can be configured independently of each other as to whether or not to use a policy service. If a policy service is used, the decisions made by the policy service replace (rather than supplement) those made by the VCS.

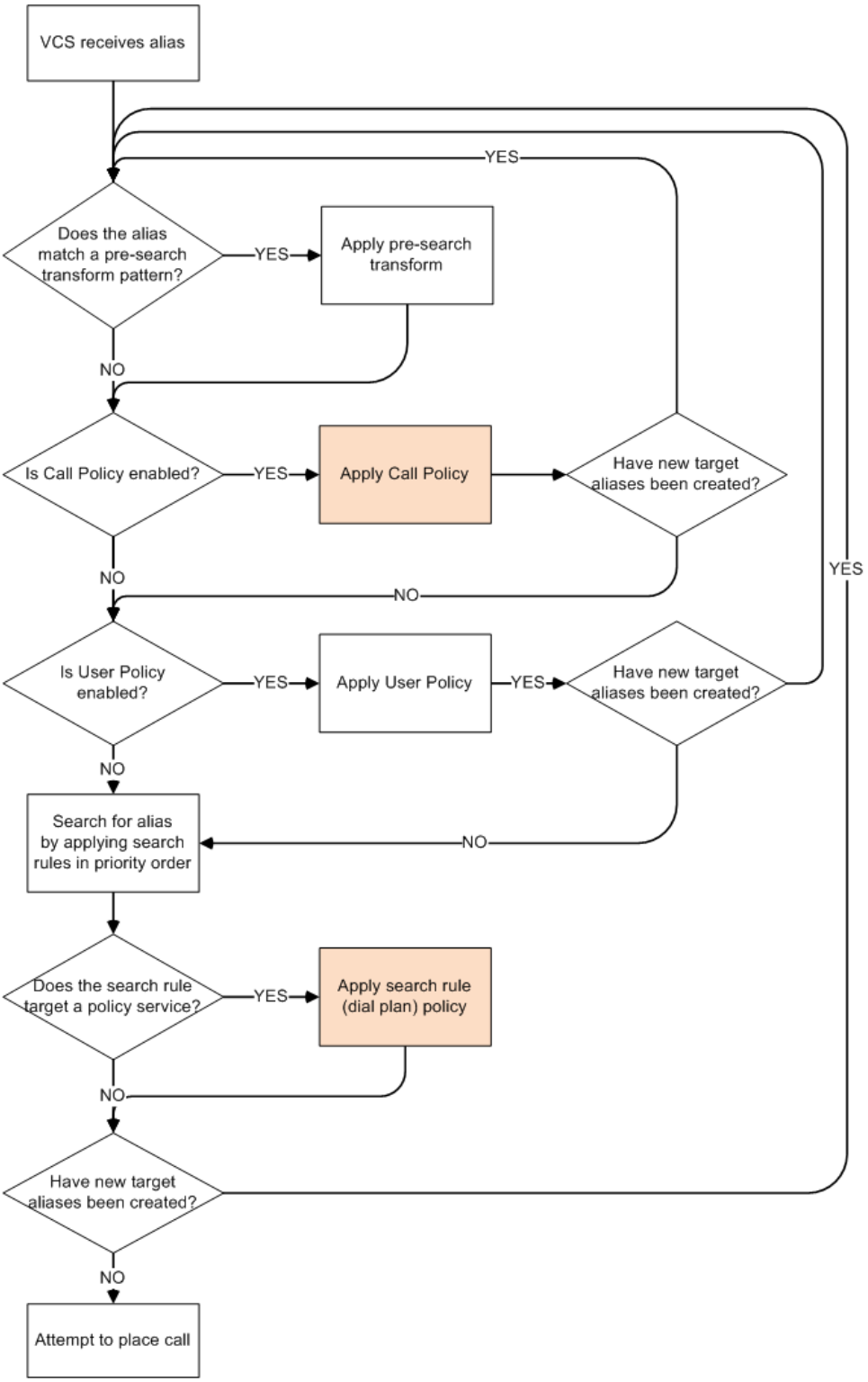
When configuring policy services:

- Up to 3 external policy servers may be specified to provide resiliency (and not load balancing).
- Default CPL can be configured, to be processed by the VCS as a fallback, if the service is not available.
- The status and reachability of the service can be queried via a status path.

If you require FindMe functionality beyond that provided by VCS / Cisco TMS, we recommend that you implement it through Call Policy.

The following flowchart shows how the VCS processes calls. The shaded boxes indicate the stages at which policy decisions can be directed to the external policy server.

Figure 1: Call processing flowchart



Whenever new locations (destination aliases) are returned in the CPL, the VCS will start its call processing from the beginning (just as it does when processing internal Call Policy CPL). Note that standard VCS loop detection protects against recursive lookups — the VCS will not search for a destination alias a second time if the call has already been proxied to that same alias.

Configuring VCS to use an external policy server

The areas where the VCS can be configured to use an external policy server are:

- Registration Policy
- Call Policy (also known as Admin Policy)
- Search rules (dial plan)

Each area can be configured independently of each other as to whether or not to use an external policy service. If an external policy service is used for a specific policy decision, the decision made by the policy service replaces (rather than supplements) the policy decision that would have been made by the VCS.

Configuring Registration Policy to use an external service

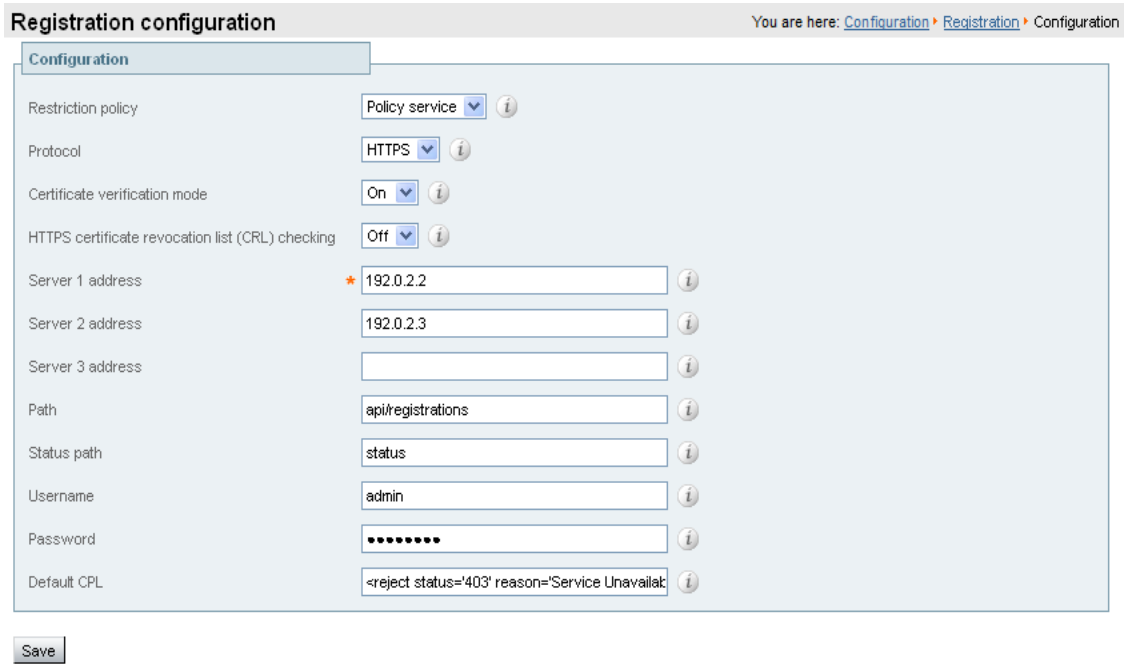
To configure Registration Policy to refer all registration restriction policy decisions out to an external service:

1. Go to **Configuration > Registration > Configuration**.
2. Select a **Restriction policy** of *Policy service*.
3. Configure the fields as follows:

Protocol	The protocol used to connect to the policy service. The default is *HTTPS*.	The VCS automatically supports HTTP to HTTPS redirection when communicating with the policy service server.
Certificate verification mode	When connecting over HTTPS, this setting controls whether the certificate presented by the policy server is verified. If *On*, for the VCS to connect to a policy server over HTTPS, the VCS must have a root CA certificate loaded that authorizes that server's server certificate. Also the certificate's Subject Common Name or Subject Alternative Name must match one of the **Server address** fields below.	The VCS's root CA certificates are loaded via (**Maintenance > Security certificates > Trusted CA certificate**).
HTTPS certificate revocation list (CRL) checking	Enable this option if you want to protect certificate checking using CRLs and you have manually loaded CRL files, or you have enabled automatic CRL updates.	Go to **Maintenance > Security certificates > CRL management** to configure how the VCS uploads CRL files.
Server address 1 - 3	Enter the IP address or Fully Qualified Domain Name (FQDN) of the server hosting the service. You can specify a port by appending `:<port>` to the address.	If an FQDN is specified, ensure that the VCS has an appropriate DNS configuration that allows the FQDN to be resolved. For resiliency, up to three server addresses can be supplied.
Path	Enter the URL of the service on the server.	

Status path	The **Status path** identifies the path from where the VCS can obtain the status of the remote service. The default is *status*.	The policy server must supply return status information, see Policy server status and resiliency [p.11].
Username	The username used by the VCS to log in and query the service.	
Password	The password used by the VCS to log in and query the service.	The maximum plaintext length is 30 characters (which is subsequently encrypted).
Default CPL	This is the fallback CPL used by the VCS if the service is not available.	You can change it, for example, to redirect to an answer service or recorded message. For more information, see Default CPL for policy services [p.10].

4. Click **Save**.
   The VCS should connect to the policy service server and start using the service for Registration Policy decisions.
   Any connection problems will be reported on this page. Check the **Status** area at the bottom of the page and check for additional information messages against the **Server address** fields.



Configuring Call Policy to use an external service

To configure Call Policy to refer all policy decisions out to an external service:

1. Go to **Configuration > Call policy > Configuration**.
2. Select a **Call Policy mode** of *Policy service*.

3. Configure the server address and connection protocols in the same manner as for Registration Policy.
4. Click **Save**.
   The VCS should connect to the policy service server and start using the service for Call Policy decisions. Any connection problems will be reported on this page. Check the **Status** area at the bottom of the page and check for additional information messages against the **Server address** fields.

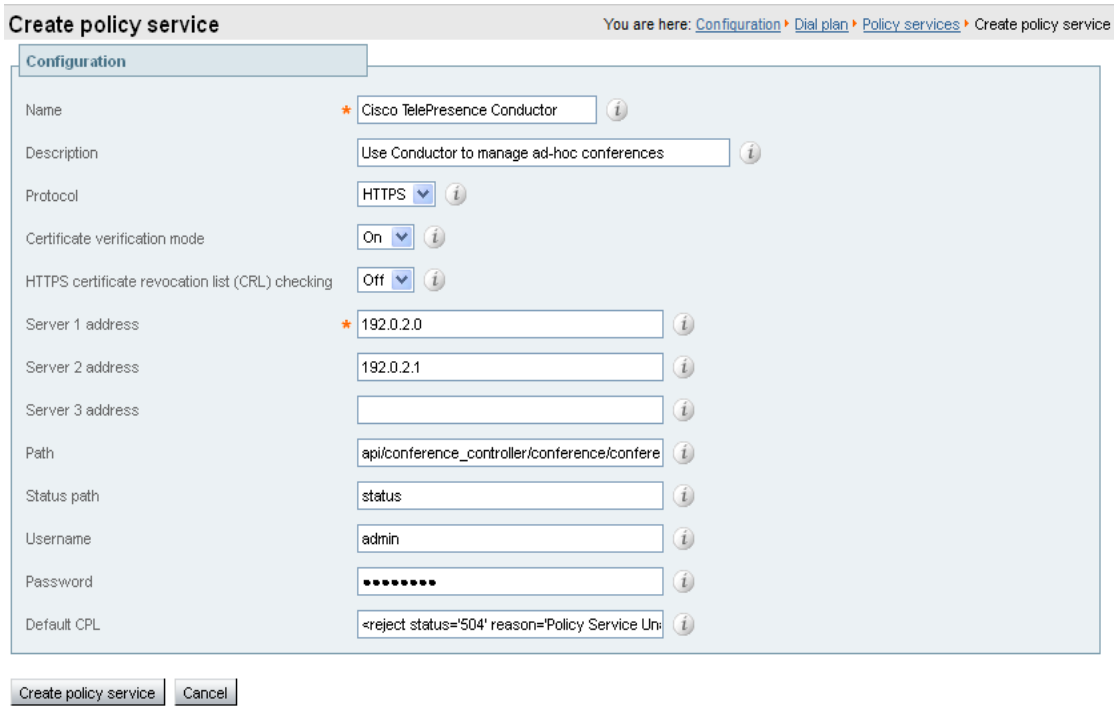
Configuring search rules to use an external service

The configuration process to set up the VCS to use an external policy service for search rules (dial plan) is broken down into the following steps:

- Configure the policy service to be used by search rules.
- Configure the relevant search rules to direct a search to the policy service.

Configuring a policy service to be used by search rules

1. Go to **Configuration > Dial plan > Policy services**.
2. Click **New**.
3. Enter a **Name** and **Description** for the policy service.
4. Configure the server address and connection protocols in the same manner as for Registration Policy.
5. Click **Create policy service**.



Configuring a search rule to direct a search to the policy service

1. Go to **Configuration > Dial plan > Search rules**.
2. Click **New**.
3. Configure the fields on the **Create search rule** page as appropriate for the searches you want to direct to the external policy server.

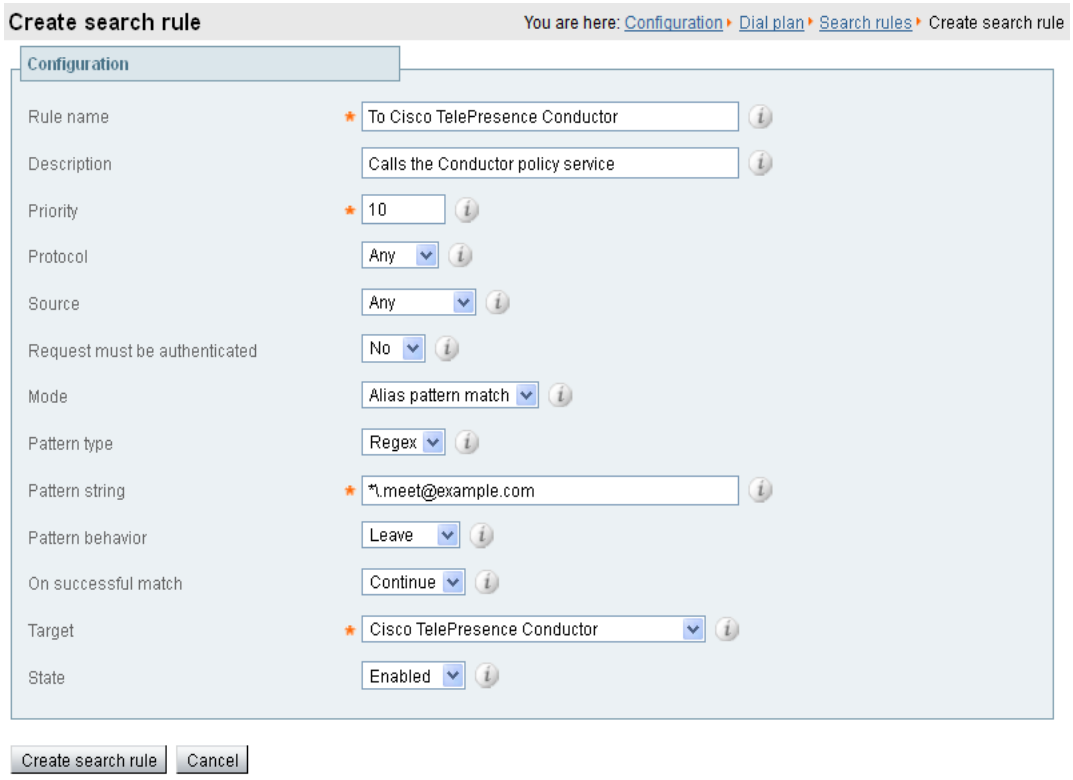
This example shows how to divert calls to aliases ending in .meet to the external policy server:

Rule name	A short name that describes the rule.
Description	A free-form description of the rule.
Priority	As required, for example 10.
Protocol	As required, for example *Any*.
Source	As required, for example *Any*.
Request must be authenticated	Configure this setting according to your authentication policy.
Mode	As required, for example *Alias pattern match*.
Pattern type	As required, for example *Regex*.
Pattern string	As required, for example.*\.meet@example.com
Pattern behavior	As required, for example *Leave*.
On successful match	As required. Note that if *Stop* is selected the VCS will not process any further search rules for the original alias, but will restart the full call processing sequence if any new aliases are returned in the CPL.
Target	Select the policy service that was created in the previous step.
State	*Enabled*

To divert all searches to the policy server you could set up 2 search rules that both target the policy service:

- The first search rule with a **Mode** of *Any alias*.
- The second search rule with a **Mode** of *Any IP address*.

4. Click **Create search rule**.



The VCS will direct all searches that match the specified pattern to the policy service server.

Your search rules must be configured in such a way that they will result in a match for the initial alias, and then either not match or not return a reject for any aliases to which the policy server has routed the call.

Default CPL for policy services

When configuring a policy service, you can specify the **Default CPL** that is used by the VCS if the service is not available.

The **Default CPL** for registrations and Call Policy defaults to:

```
<reject status='403' reason='Service Unavailable'/>
```

and this will reject the request.

The **Default CPL** for policy services used by search rules defaults to:

```
<reject status='504' reason='Policy Service Unavailable'/>
```

and this will stop the search via that particular search rule.

This default CPL mean that in the event of a loss of connectivity to the policy server, all call and registration requests will be rejected. If this is not your required behavior then you are recommended to specify alternative default CPL.

We recommend that you use unique reason values for each type of service, so that if calls or registrations are rejected it is clear why and which service is rejecting the request.

Policy server status and resiliency

You must specify a **Status path** when configuring the VCS's connection to a policy server. It identifies the path from where the status of the remote service can be obtained. By default this is *status*.

Up to 3 different policy server addresses may be specified. The VCS polls each address on the specified path every 60 seconds to test the reachability of that address. The VCS accepts standard HTTP(S) response status codes. (Note that the developers of the policy service must ensure that this provides the appropriate status of the service.)

If a server does not respond to status requests, VCS will deem that server's status to be in a failed state and it will not be queried for policy service requests until it returns to an active state. Its availability will not be checked again until after the 60 second polling interval has elapsed.

When the VCS needs to make a policy service request, it attempts to contact the service via one of the configured server addresses. It will try each address in turn, starting with **Server 1 address**, and then if necessary - and if configured - via the **Server 2 address** and then the **Server 3 address**. The VCS only tries to use a server address if it is in an active state, based on its most recent status query.

The VCS has a non-configurable 30 seconds timeout value for each attempt it makes to contact a policy server. However, if the server is not reachable, the connection failure will occur almost instantaneously. (Note that the TCP connection timeout is usually 75 seconds. Therefore, in practice, a TCP connection timeout is unlikely to occur as either the connection will be instantly unreachable or the 30 second request timeout will occur first.)

The VCS uses the configured **Default CPL** if it fails to contact the policy service via any of the configured addresses.

Note that this method provides resiliency but not load balancing i.e. all requests will be sent to **Server 1 address**, providing that server address is functioning correctly.

Viewing policy server status via the VCS

A summarized view of the status of the connection to each policy service can be viewed by going to the **Policy service status** page (**Status > Policy services**).

The set of policy services includes all of the services defined on the **Policy services** page (**Configuration > Dial plan > Policy services**), plus if a remote service has been selected for either Call Policy or for registration restriction policy it will also display a **Call Policy** or a **Registration restriction** service respectively.

The following information is displayed:

Field	Description
Name	The name of the policy service. Clicking on a **Name** takes you to the configuration page for that service where you can change any of the settings or see the details of any connection problems.
URL	The address of the service. Note that each service can be configured with multiple server addresses for resiliency. This field displays the server address currently selected for use by the VCS.
Status	The current status of the service based on the last attempt to poll that server.
Last used	Indicates when the service was last requested by the VCS.

External policy request parameters

When the VCS uses a policy service it sends information about the call or registration request to the service in a POST message using a set of name-value pair parameters. The service can then make decisions based upon these parameters combined with its own policy decision logic and supporting data (for example lists of aliases that are allowed to register or make and receive calls, via external data lookups such as an LDAP database or other information sources).

The service response must be a 200 OK message with CPL contained in the body.

The following table lists the possible parameters contained within a request and indicates with a ✓ in which request types that parameter is included. It also indicates, where relevant, the range of accepted values.

Parameter name	Values	Registration Policy	Search rules	Call Policy	User Policy
ALIAS		✓			
ALLOW_INTERWORKING	TRUE / FALSE		✓	✓	✓
AUTHENTICATED	TRUE / FALSE	✓	✓	✓	✓
AUTHENTICATED_SOURCE_ALIAS			✓	✓	✓
AUTHENTICATION_USER_NAME			✓	✓	✓
CLUSTER_NAME		✓	✓	✓	✓
DESTINATION_ALIAS			✓	✓	✓
DESTINATION_ALIAS_PARAMS			✓	✓	✓
GLOBAL_CALL-SERIAL_NUMBER	GUID		✓	✓	✓
LOCAL_CALL_SERIAL_NUMBER	GUID		✓	✓	✓
METHOD	INVITE / ARQ / LRQ / OPTIONS / SETUP / REGISTER / SUBSCRIBE / PUBLISH	✓	✓	✓	✓
NETWORK_TYPE	IPV4 / IPV6		✓	✓	✓
POLICY_TYPE	REGISTRATION / SEARCH / ADMIN / USER	✓	✓	✓	✓
PROTOCOL	SIP / H323	✓	✓	✓	✓
REGISTERED_ALIAS			✓	✓	✓
SOURCE_ADDRESS		✓	✓	✓	✓
SOURCE_IP		✓	✓	✓	✓
SOURCE_PORT		✓	✓	✓	✓

Parameter name	Values	Registration Policy	Search rules	Call Policy	User Policy
TRAVERSAL_TYPE	TYPE_[UNDEF / ASSENTSERVER / ASSENTCLIENT / H460SERVER / H460CLIENT / TURNSERVER / TURNCLIENT / ICE]		✓	✓	✓
UNAUTHENTICATED_SOURCE_ALIAS			✓	✓	✓
UTCTIME		✓	✓	✓	✓
ZONE_NAME			✓	✓	✓

Appendix 2: CPL snippet examples [p.23] contains some examples of the types of CPL that the policy server could use in its response.

Cryptography support

External policy servers should support TLS and AES-256/AES-128/3DES-168.

SHA-1 is required for MAC and Diffie-Hellman / Elliptic Curve Diffie-Hellman key exchange; the VCS does not support MD5.

Appendix 1: Design examples

Call Policy design examples

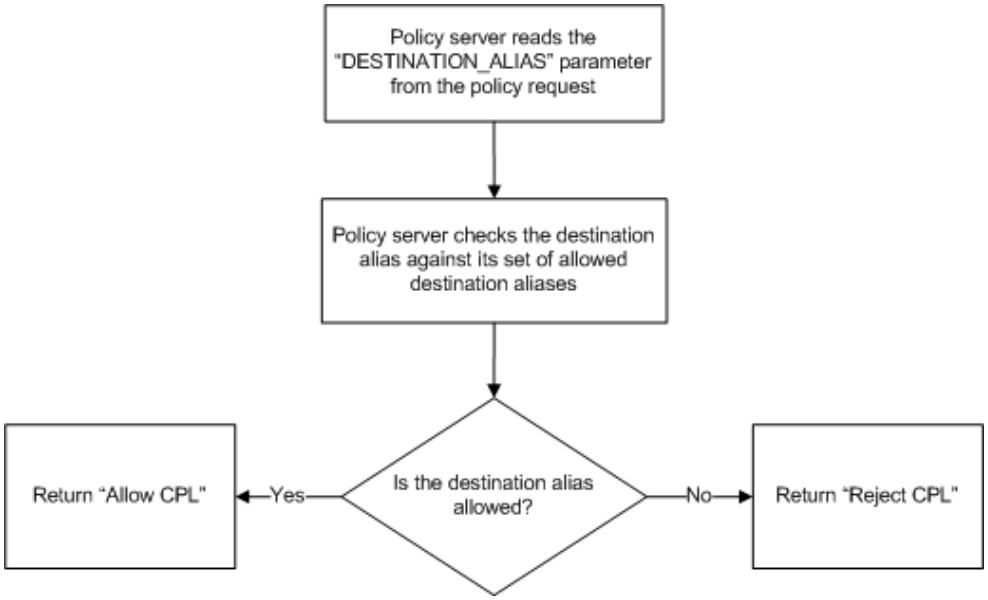
This section provides some flowchart examples of how a policy service could implement policy rules.

The examples refer to types of CPL such as “Allow CPL” or “Reject CPL” that would be returned by the policy service (see Appendix 2: CPL snippet examples [p.23] for examples of the actual CPL that could be returned).

Using a policy service to allow or deny calls

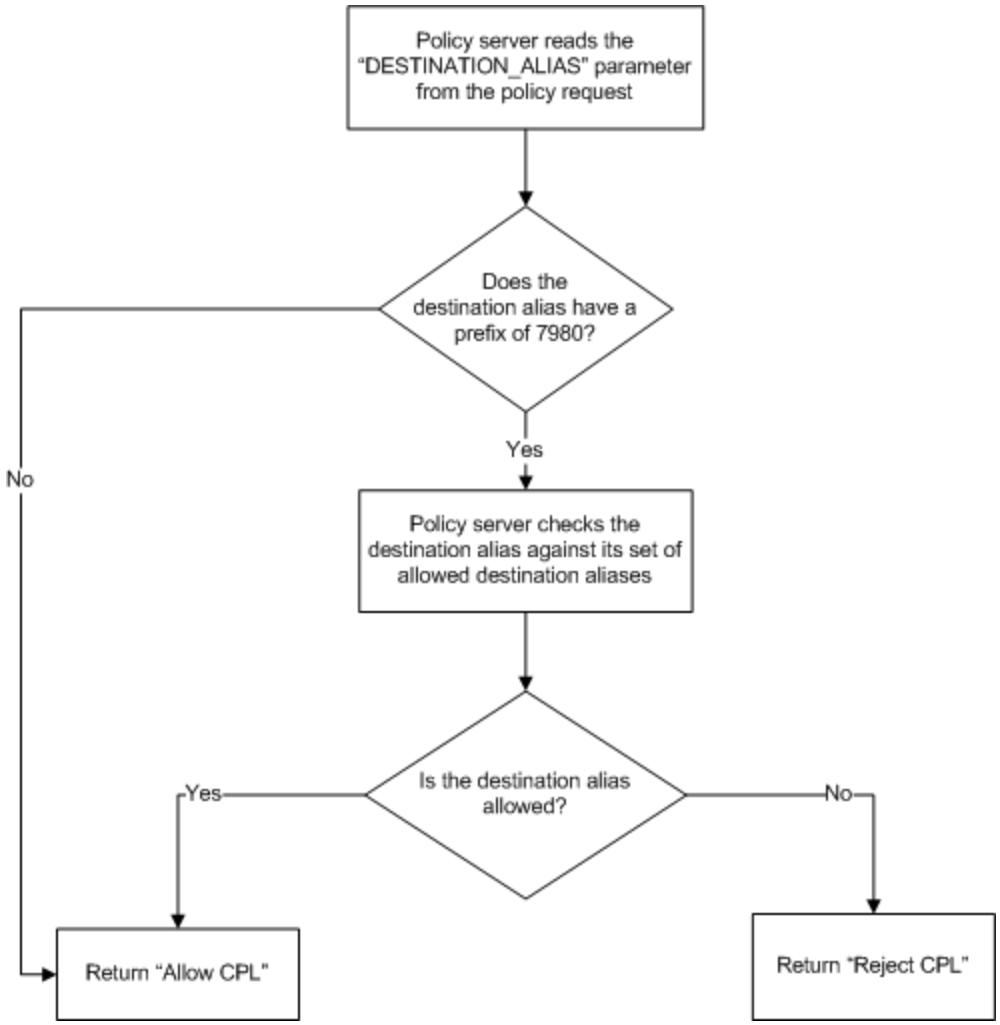
Destination alias whitelist

In this example the network administrator wants to allow only calls to approved destination aliases.



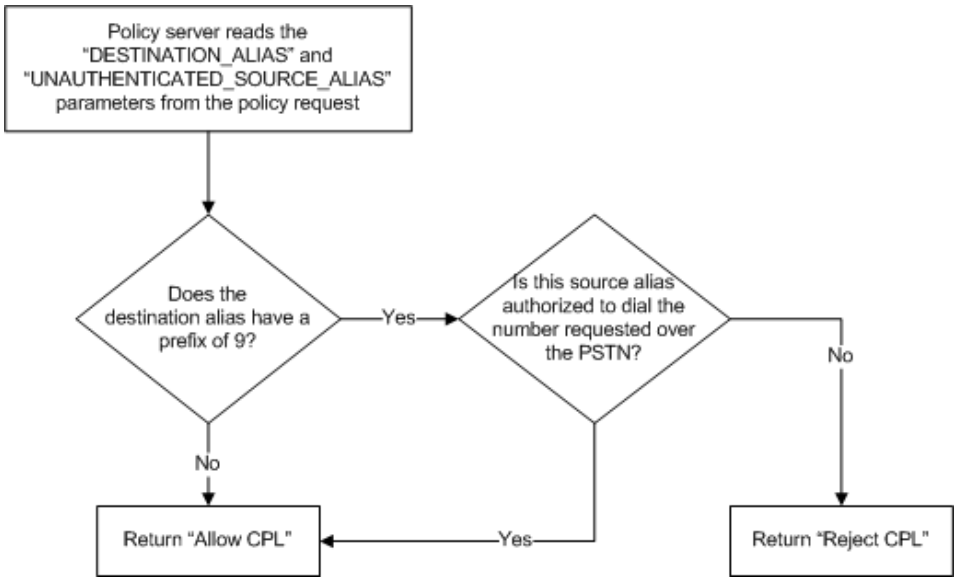
Whitelist of a subset of calls

In this example the network administrator wants to filter calls starting with the prefix “7980”, all other calls should be allowed. In this case it is necessary to include an implicit allow rule for most dialed aliases and to only filter destination aliases that match a prefix.



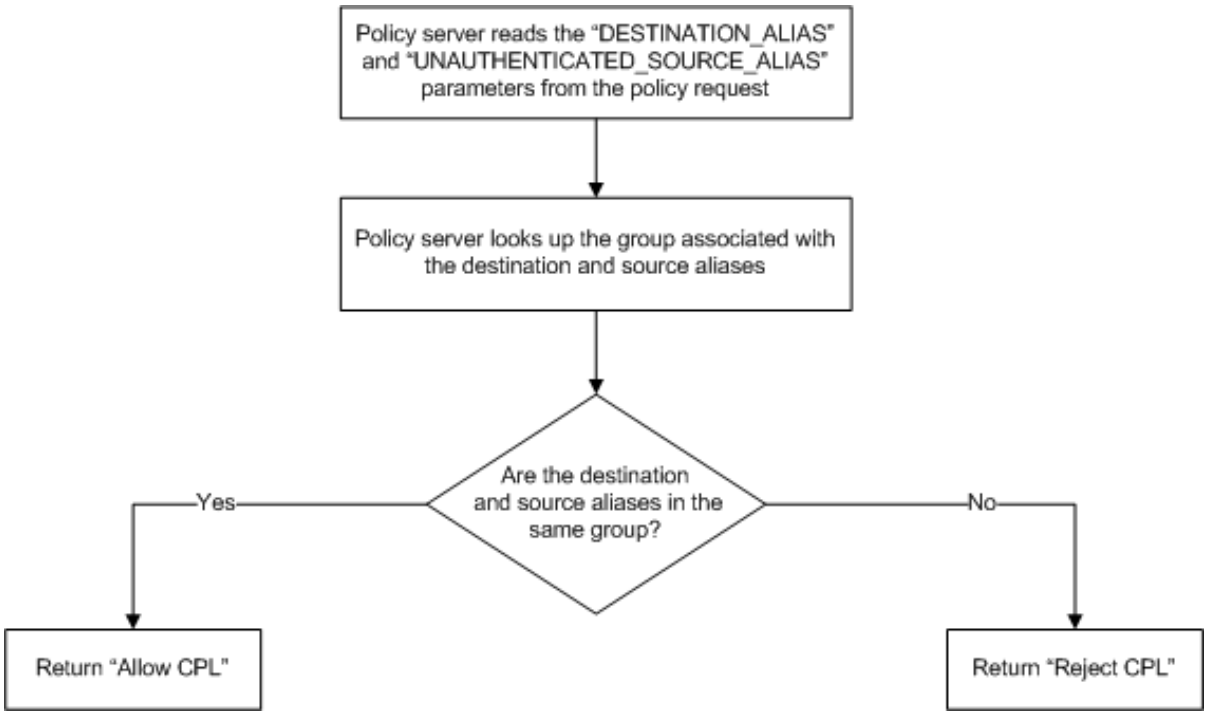
Whitelisting calls based on user privileges

In this example an administrator wants to limit each user's ability to dial out on to the PSTN according to their privilege (determined by their source alias). This example assumes that users dial out onto the PSTN by prefixing their dialed number with a "9".



Intra-group calling

In this example the policy server is managing the video network for multiple companies. It only allows each company to call other endpoints in their company. It does this by setting up groups of aliases per company; each group contains only aliases that belong to the same company.



Using a policy service to route calls

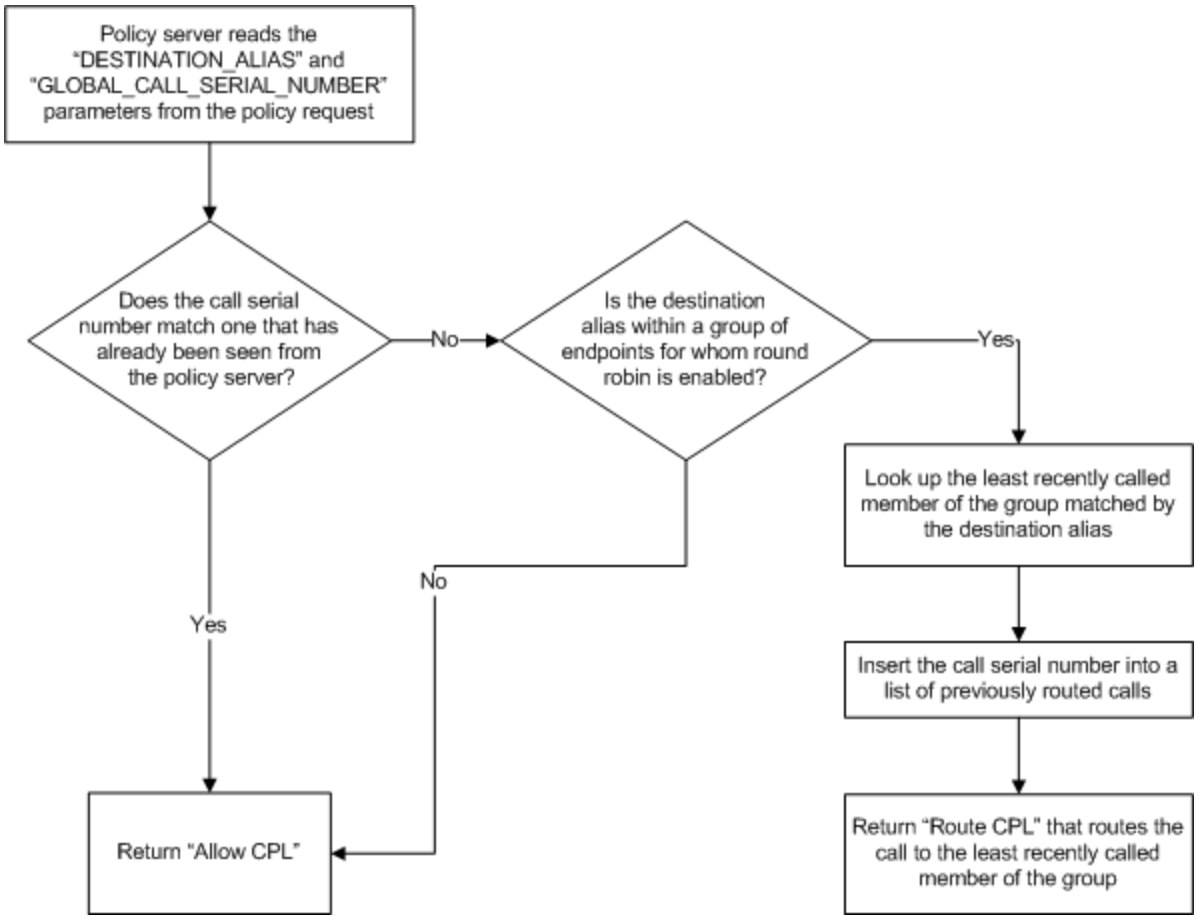
When using external policy servers to control call routing, the CPL returned to the VCS by the policy server could change the destination alias of the call or add extra destinations. In these cases, the VCS will make another request to the policy server for the new or modified destination aliases.

This may be a desirable feature in some scenarios, but often in order to save resources it will not be desirable for the policy server to fully process calls it has already routed, especially in the case of routing and forking where loops or excessive forking could occur.

To assist in managing these scenarios, the "GLOBAL_CALL_SERIAL_NUMBER" can be used to identify calls that the policy server has already processed. This value is unique per call, across all VCSs.

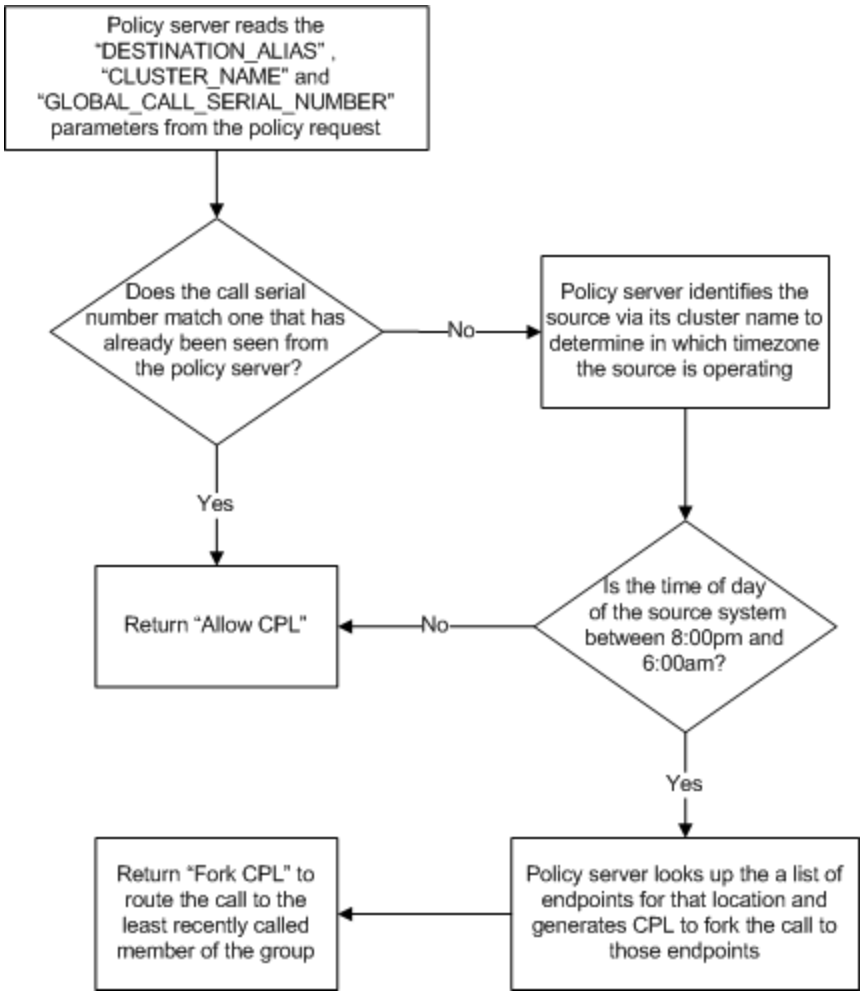
Round-robin routing to a member of a group

In this example an administrator wants to route calls to the member of a group who has least recently received a call. This requires an alias which represents members of a group and a list of the members within that group. The administrator still wants to retain the ability for users to call other users directly.



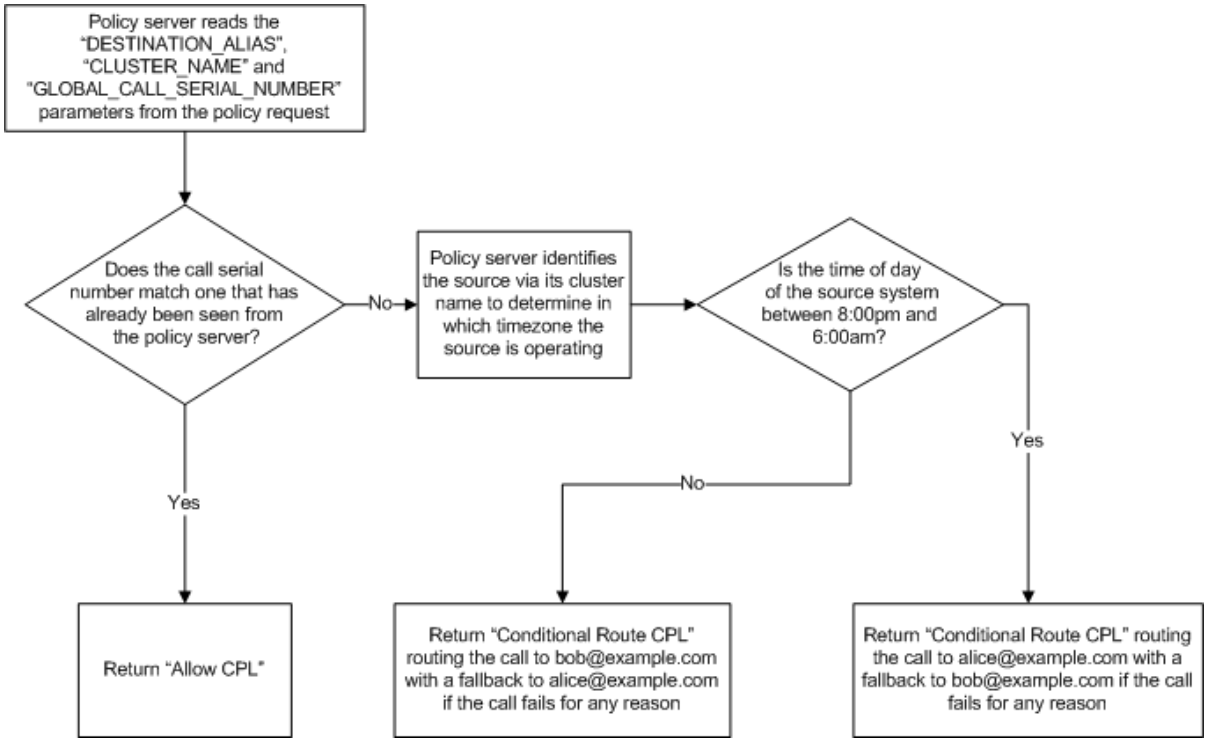
Forking calls based on time of day

In this example an administrator wants to enable a “night mode” whereby if a phone call arrives after 8:00PM and before 6:00AM the call is routed to multiple endpoints to increase the chance of someone picking up the call.



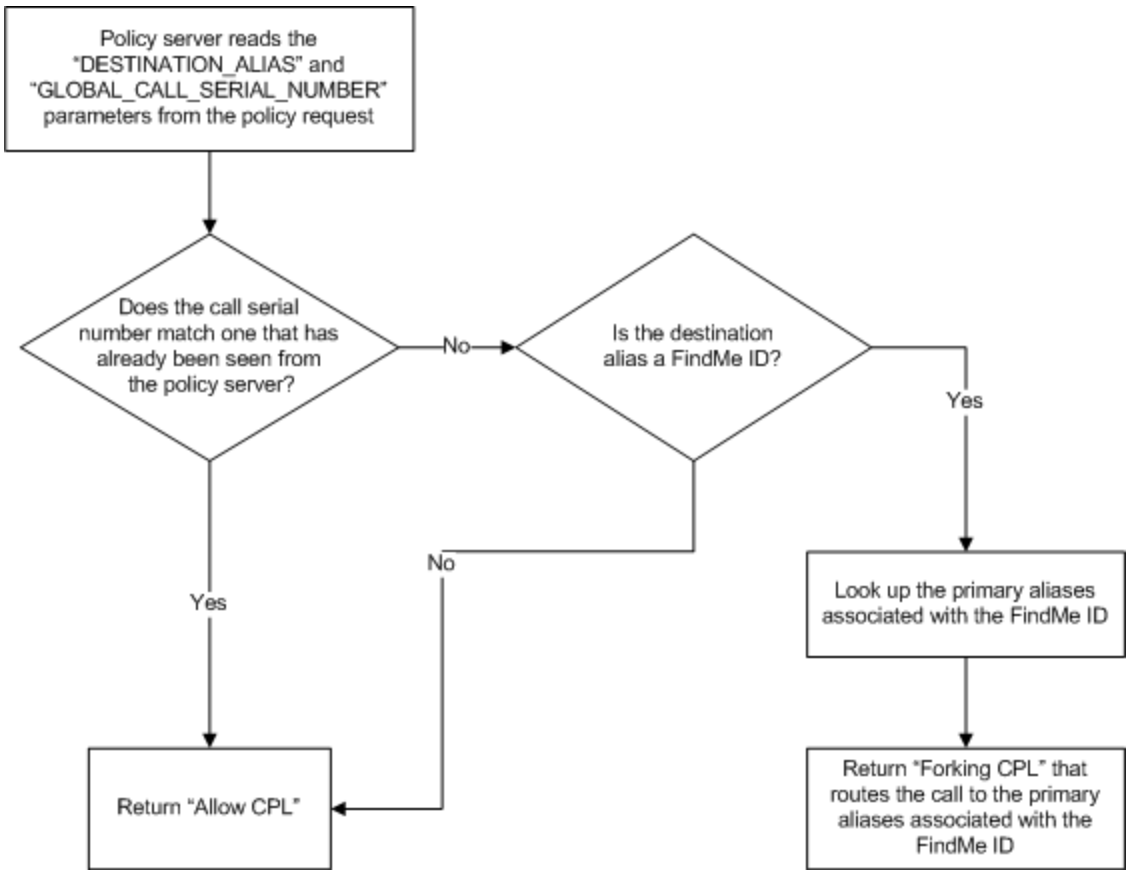
Routing calls conditionally with failovers

In this example an administrator wants to route a call to two different users dependent on time of day with a failover to the other user if the first fails to answer.



Using a policy service to implement FindMe (User Policy)

In this example the destination alias is checked to see if it is a FindMe ID. If so, the server looks up the aliases associated with that FindMe and forks the call to those aliases.



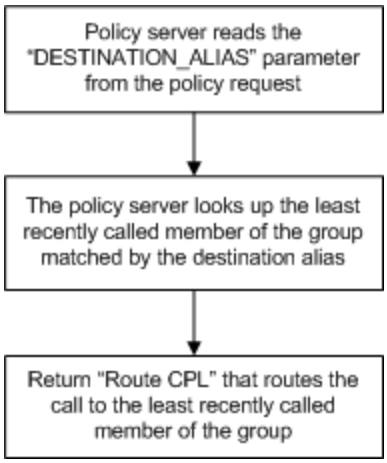
Search rule design examples

Policy services used by search rules are similar to Call Policy services; however they allow you to use VCS search rules to filter which calls are directed to the policy services.

Round-robin routing to a member of a group

In this example an administrator wants to route calls to the member of a group who has least recently received a call. This requires an alias which represents members of a group and a list of the members within that group. The administrator still wants to retain the ability for users to call other users directly.

In this case the search rule is configured to match only aliases for which round robin groups on the external policy server are present.

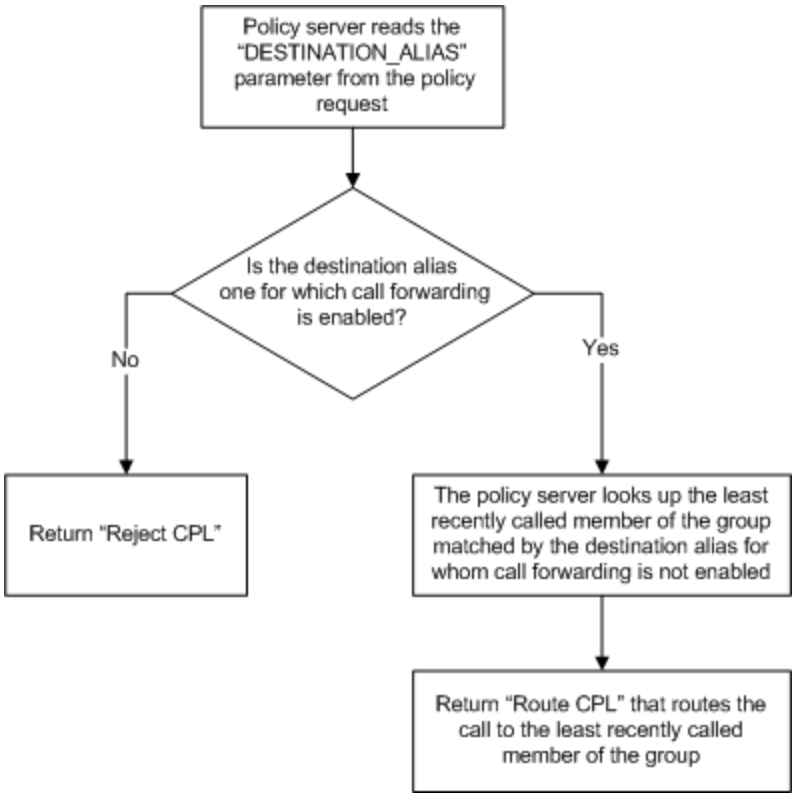


Forwarding calls to other members of a group in a round-robin style

In this example calls to a member of a team who is currently unavailable to take calls (for example they are on leave or in a location without video), are redirected to another member of the team.

It requires the external policy server to know who is unavailable and who the other team members are.

In this case a search rule is configured to match the range of destination aliases for which the administrator may want to redirect calls.



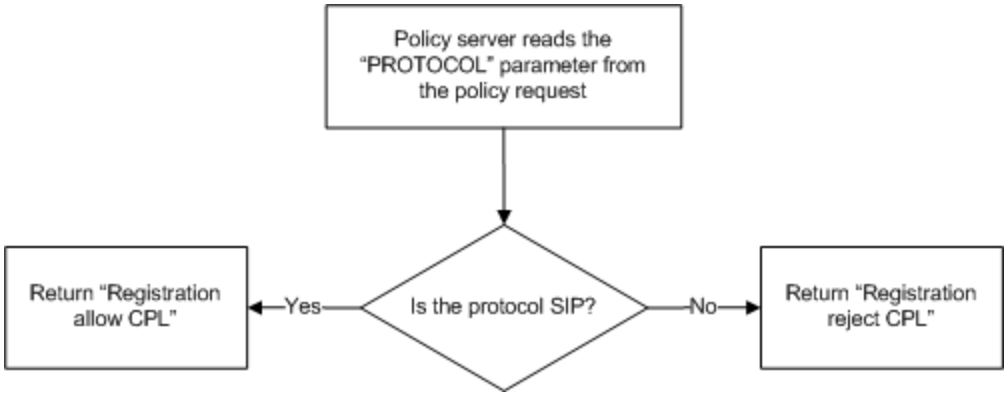
The search rule that routes the redirected call specified by the policy server must be at a lower priority than that of the policy service check search rule.

- When the VCS receives a "Route CPL" message from the external policy service it stops its current search and starts a fresh search with the new location (destination).
- When the VCS receives a reject message from a policy service configured in the search rules it fails this search rule but then continues the search and works through the lower priority search rules.

Registration Policy design examples

Allowing or denying registrations based on protocol

In this example an administrator wants the VCS to accept only SIP registrations but to be capable of routing H.323 calls (both SIP and H.323 must be enabled on the VCS).



Appendix 2: CPL snippet examples

This section contains examples of CPL snippets that can be returned by an external policy service to the VCS.

CPL snippets for call processing

Allow CPL

This CPL can be used to allow a call to proceed:

```
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:taa="http://www.tandberg.net/cpl-extensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
  <taa:routed>
    <!-- Route call but clear after 30 seconds if no answer -->
      <proxy timeout="30"/>
  </taa:routed>
</cpl>
```

Reject CPL

This CPL can be used to reject a call and supply a reject reason:

```
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:taa="http://www.tandberg.net/cpl-extensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
  <taa:routed>
    <!-- Reject call with reason 403 (SIP Forbidden Code) and message-->
    <reject status="403" reason="Alias not in allowed list"/>
  </taa:routed>
</cpl>
```

Route CPL

This CPL can be used to unconditionally redirect a call:

```
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:taa="http://www.tandberg.net/cpl-extensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
  <taa:routed>
    <!--Redirect the call to alice@example.com by clearing the
    current list of destination aliases through (clear=yes)
    and adding a new alias (url=alice@example.com)-->
    <taa:location clear="yes" url="alice@example.com">
      <proxy/>
    </taa:location>
  </taa:routed>
</cpl>
```

Forking CPL

This CPL can be used to fork a call to multiple aliases:

```
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:taa="http://www.tandberg.net/cpl-extensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
  <taa:routed>
    <!--Fork the call to endpoint1@example.com and add new aliases
        endpoint2@example.com and endpoint3@example.com -->
    <taa:location clear="no" url="endpoint1@example.com">
      <!--Fork the call to a second alias (endpoint2@example.com)-->
      <taa:location url="endpoint2@example.com">
        <!--Fork the call to a third alias (endpoint3@example.com)-->
        <taa:location url="endpoint3@example.com">
          <proxy/>
        </taa:location>
      </taa:location>
    </taa:location>
  </taa:routed>
</cpl>
```

Conditional routing CPL

This CPL can be used to redirect a call under specific conditions. In this example, if a call that initially routes to alice is not answered then the call redirects to bob:

```
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:taa="http://www.tandberg.net/cpl-extensions"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
  <taa:routed>
    <!--Clear the destination aliases (clear=yes)and
    add the destination alias alice@example.com
    url="alice@example.com")-->
    <taa:location clear="yes" url="alice@example.com">
      <proxy timeout="10">
        <!-- If the call setup fails for any reason or takes more than ten seconds
        to complete then the CPL within the default tag is activated -->
        <default>
          <!--Clear the destination aliases (clear=yes) and add the
          destination alias bob@example.com (url="bob@example.com")-->
          <taa:location clear="yes" url="bob@example.com">
            <proxy/>
          </taa:location>
        </default>
      </proxy>
    </taa:location>
  </taa:routed>
</cpl>
```

CPL snippets for registration requests

Note that the following example CPL snippets for registration requests are more limited than the other example snippets. This is because the VCS does not support the full range of CPL for registration requests.

It only looks for <proxy/> or <reject/> tags and ignores any other content.

Registration allow CPL

This CPL can be used to accept a registration request:

```
<proxy/>
```

Registration reject CPL

This CPL can be used to reject a registration request and supply a reject reason:

```
<reject status="403" reason="H323 registrations not allowed"/>
```

Appendix 3: Message logging

You can monitor the policy request messages and responses that are exchanged between the VCS and the policy service.

The best way to do this is to use the diagnostic logging tool to capture these messages:

1. Go to **Maintenance > Diagnostics > Diagnostic logging**.
2. Optionally, select **Take tcpdump while logging**.
3. Click **Start new log**.
4. (Optional) Enter some **Marker** text and click **Add marker**.
   - The marker facility can be used to add comment text to the log file before certain activities are performed. This helps to subsequently identify the relevant sections in the downloaded diagnostic log file.
   - You can add as many markers as required, at any time while the diagnostic logging is in progress.
   - Marker text is added to the log with a "**DEBUG_MARKER**" tag.
5. Reproduce the system issue you want to trace in the diagnostic log.
6. Click **Stop logging**.
7. Click **Download log** to save the diagnostic log to your local file system. You are prompted to save the file (the exact wording depends on your browser).
8. If appropriate, click **Download tcpdump** to also download the tcpdump file to your local file system.

Trace example: Call Policy request and response

Example Call Policy request:

```
Jul 19 15:30:30 vcs tvcs: UTCTime="2011-07-19 15:30:30,616" Module="network.http"
Level="DEBUG":  Message="Request" Method="POST" URL=" https://192.0.2.3/api/call_policy"
Ref="0x4945360"
Data="ALLOW_INTERWORKING=TRUE&AUTHENTICATED=FALSE&AUTHENTICATION_USER_NAME=&CLUSTER_
NAME=vcs_cluster&DESTINATION_ALIAS=alice%40example.com&GLOBAL_CALL_SERIAL_NUMBER=094f761
c-b21c-11e0-91a2-000c29e127de&LOCAL_CALL_SERIAL_NUMBER=094f754a-b21c-11e0-a091-000c29e127
de&METHOD=INVITE&NETWORK_TYPE=IPV4&POLICY_TYPE=ADMIN&PROTOCOL=SIP&REGISTERED_ALIAS=bob%40
example.com&SOURCE_ADDRESS=192.0.2.100%3A5061&SOURCE_IP=192.0.2.100&SOURCE_PORT=5061&TRAV
ERSAL_TYPE=TYPE_UNDEF&UNAUTHENTICATED_SOURCE_ALIAS=bob%40example.com&UTCTIME=2011-07-19%2
015%3A30%3A30&ZONE_NAME=DefaultSubZone"
```

Example response:

```
Jul 19 15:30:30 vcs tvcs: UTCTime="2011-07-19 15:30:30,625" Module="network.http"
Level="DEBUG":  Message="Response" Src-ip="192.0.2.3" Src-port="5000" Dst-ip="192.0.2.20
0" Dst-port="40010" Response="200 OK" ResponseTime="0.003416"
Body="<!-- policy server -->
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
xmlns:taa="http://www.tandberg.net/cpl-extensions"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
<taa:routed> <proxy/> </taa:routed> </cpl> " Ref="0x4945360"
```

Trace example: Registration Policy request and response

Example Registration Policy request:

```
Jul 26 13:36:51 vcs tvcs: UTCTime="2011-07-26 13:36:51,024" Module="network.http"
Level="DEBUG":  Message="Request" Method="POST"
URL="http://192.0.2.3:5000/custompolicy.cpl" Ref="0x7f58105481c0"
Data="ALIAS=bob%40example.com&AUTHENTICATED=FALSE&AUTHENTICATION_USER_NAME=&CLUSTER_NAME=
vcs_cluster&METHOD=REGISTER&NETWORK_TYPE=IPV4&POLICY_TYPE=REGISTRATION&PROTOCOL=SIP&SOURC
E_IP=192.0.2.100&SOURCE_PORT=5061&TRAVERSAL_TYPE=TYPE_UNDEF&UTCTIME=2011-07-26%2013%3A36%
3A51&ZONE_NAME=DefaultSubZone"
```

Example “accept” response:

```
Jul 26 13:36:51 vcs tvcs: UTCTime="2011-07-26 13:36:51,031" Module="network.http"
Level="DEBUG":  Message="Response" Src-ip="192.0.2.3" Src-port="5000"
Dst-ip="192.0.2.200" Dst-port="42510" Response="200 OK" ResponseTime="0.007301"
Body="<!-- policy server-->
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
xmlns:taa="http://www.tandberg.net/cpl-extensions"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
<taa:routed> <proxy/> </taa:routed> </cpl> " Ref="0x7f58105481c0"
```

Example “reject” response:

```
Jul 26 13:36:51 vcs tvcs: UTCTime="2011-07-26 13:36:51,031" Module="network.http"
Level="DEBUG":  Message="Response" Src-ip="192.0.2.3" Src-port="5000"
Dst-ip="192.0.2.200" Dst-port="42510" Response="200 OK" ResponseTime="0.007301"
Body="<!-- policy server-->
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
xmlns:taa="http://www.tandberg.net/cpl-extensions"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd">
<taa:routed>  <reject status="403" reason="Alias not in allowed list"/>
</taa:routed>  </cpl> " Ref="0x7f58105481c0"
```

Document revision history

The following table summarizes the changes that have been applied to this document.

Revision	Date	Description
4	December 2013	Updated for X8.1.
3	April 2013	Restructured document content, and added more policy server status and resiliency information.
2	August 2012	Updated for new protocol search rule option introduced in VCS X7.2.
1	September 2011	Initial release.