Cisco IOS Voice Translations – Part 4: Testing and Troubleshooting Voice Translation Profiles

This is the fourth part of a series of four articles on Cisco IOS Voice Translations. In Part 1, I covered how to read and write simple regular expressions to construct individual translation rules. In Part 2, I covered how to combine rules into rulesets (voice translation-rules) and the rulesets into voice translation-profiles. Parts 3 took this one step farther and discussed advanced regular expressions and special applications of voice translation rules.

Now that you have the tools to build voice translation-profiles, in this article we will look at the tools you can use to test what you have built.

Here are the links to the other posts in this series that have been written so far:

- **Part 1:** http://www.sunsetlearning.com/training-resources/cisco-ios-voice-translationspart-1-the-basics-of-voice-translation-rules/
- **Part 2:** http://www.sunsetlearning.com/Cisco-IOS-Voice-Translations-Part-2-Writing-and-Applying-Voice-Translation-Rules-and-Voice-Translation-Profiles
- **Part 3:** http://www.sunsetlearning.com/training-resources/sli-blogs/cisco-ios-voice-translations-part-3-building-applying-advanced-expressions-voice-translation-profiles/

First, let's review the building blocks of voice translations in IOS:

1) Rule: A single match/replace rule. (Covered in Part 1)
2) Voice Translation Rule: A set of individual rules. A Voice Translation Rule will be used to match numbers in a called party, calling party or redirect number. Since it is a set of rules, it can be used to match one or more patterns of numbers and have each pattern manipulated in a different way.
3) Voice Translation Profile: A set of one or more Voice Translation Rules. A Voice Translation Profile allows you to indicate one Voice Translation Rule for called party, one Voice Translation Rule for calling party and one Voice Translation Rule for redirect number.
4) Application of a Voice Translation Profile: Profiles are most commonly applied to ports and dial-peers. When a profile is applied, a direction is declared. One profile can be applied to calls inbound to the port or dial-peer, and another can be applied to calls that are processed through that same port or dial-peer when flowing outbound.

The first two tools we will look at are simple "show" commands: "**show voice translation-profile**" and "**show voice translation-rule**". Let's look at a voice translation profile and then look at the show command output. The following voice translation-profile was built in Part 3 of this series, so hopefully you are familiar with it:

```
voice translation-rule 1
       rule 1  /7035557/  /7/
       rule 2  /9725553/  /3/
       rule 3  /4065559/  /5/

voice translation-rule 2
       rule 1  /.*/  /9&/  type  subscriber  subscriber
       rule 2  /.*/  /91&/  type  national  national
       rule 3  /.*/  /9011&/  type  international  international
       rule 4  /.*/  /9&/  type  unknown  unknown

voice translation-profile pstn-in
       translate called 1
       translate calling 2
```

Here is the output of the "show voice translation-profile command:

```
show voice translation-profile
Translation Profile: pstn-in
       Rule for Calling number: 2
       Rule for Called number: 1
       Rule for Redirect number:
       Rule for Redirect-target number:
```

That output is pretty straightforward. The "show voice translation-profile" without specifying which profile we want to see would list all of the profiles in the router. If that is too long an output, we could refine it with "show voice translation-profile pstn-in".

Next let's look at the output of "show voice translation-rule":

```
show voice translation-rule 1
Translation-rule tag: 1

        Rule 1:
        Match pattern: 7035557
        Replace pattern: 7
        Match type: none                Replace type: none
        Match plan: none                Replace plan: none

        Rule 2:
        Match pattern: 9725553
        Replace pattern: 3
        Match type: none                Replace type: none
        Match plan: none                Replace plan: none

        Rule 3:
        Match pattern: 4065559
        Replace pattern: 5
        Match type: none                Replace type: none
        Match plan: none                Replace plan: none
```

Note that each rule in the ruleset is shown with its match/replace pattern as well as any match/replace rules for type of number (TON) or numbering plan. As with "show voice translation-rule" we can leave the argument off and see all of the voice translation rules in the router, or we can specify which ruleset we want to see.

Now things will get more interesting. What if you have a complex ruleset and want to see what rule is matched, and what the resulting number will be? The command "**test voice translation-rule**" allows you to test your rulesets.

```
test voice translation-rule <translation-rule number> <input string>

test voice translation-rule 1 4065559123
Matched with rule 3
Original number: 4065559123     Translated number: 5123
Original number type: none      Translated number type: none
Original number plan: none      Translated number plan: none
```

What about voice translation-rules that include TON or numbering plan? They can be tested as well, but because the TON or numbering plan is part of the match/replace rule a TON or numbering plan must be part of your tested number. For example:

```
test voice translation-rule 2 7035554444 type national
Matched with rule 2
Original number: 7035554444     Translated number: 917035554444
Original number type: national  Translated number type: national
Original number plan: none      Translated number plan: none
```

What happens if the number that you are testing does not match any rule in the ruleset? In that case, the router would simply return a command prompt without (unfortunately) any error message. The output would look like this:

```
HQ-1#
HQ-1#test voice translation-rule 1 5715551234
HQ-1#
```

Another useful command when working with voice translations is the "show dialplan number"* command. In addition to showing you what dial-peer will be matched and which, if any, voice translation profiles will be used:

```
show dialplan incall 0/1/0:23 number 15155552001
Matched port

VoiceEncapPeer 7
    peer type = voice, system default peer = FALSE, information type = voice,
    description = `Inbound to HQ',
    tag = 7, destination-pattern = `15155552...$',
    .
    .
    .
    Translation profile (Incoming):pstn-in
    Translation profile (Outgoing):
    .
    .
    .
```

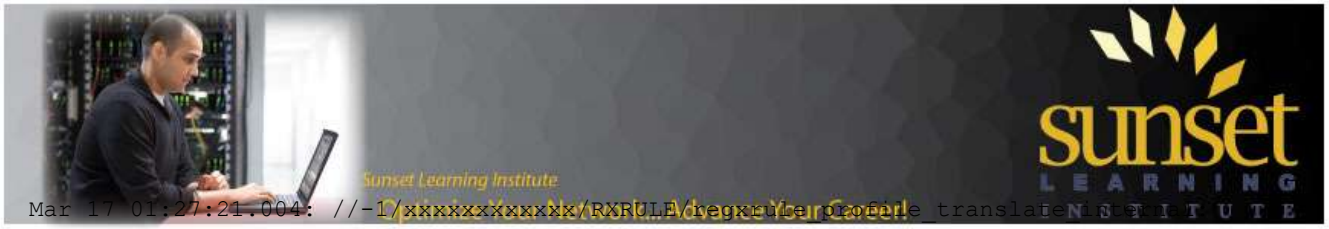
* For more information on the "show dialplan" command, see the "Cisco IOS Voice Command Reference Guide".

Finally, you can debug voice translations with the appropriately named "debug voice translation" command. In order to generate output on an idle router, use a command like "csim start" to generate input/output:

```
HQ-1#
HQ-1#debug voice translation
VoIP Translation Rule debugging is enabled
HQ-1#
HQ-1#
HQ-1#csim start 9725553001
csim: called number = 9725553001, loop count = 1 ping count = 0

csim: loop = 1, failed = 1
csim: call attempted = 1, setup failed = 1, tone failed = 0

HQ-1#
Mar 17 01:27:21.004: //-
1/xxxxxxxxxxxx/RXRULE/regxrule_get_profile_from_voiceport_internal: Found profile
pstn-in defined on voice-port
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal:
number= type=unknown plan=unknown numbertype=calling
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x4
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x2
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x1
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_match_internal: Matched
with rule 4 in ruleset 2
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x4
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x2
Mar 17 01:27:21.004: //-1
HQ-1#/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't match; in.type=0x0
rule.type = 0x1
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_match_internal: Matched
with rule 4 in ruleset 2
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x4
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x2
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: Error: type didn't
match; in.type=0x0 rule.type = 0x1
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/sed_subst: Successful substitution;
pattern= matchPattern=.* replacePattern=9& replaced pattern=9
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_subst_num_type: Match Type =
unknown, Replace Type = unknown Input Type = unknown
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_subst_num_plan: Match Plan =
none, Replace Plan = none Input Plan = unknown
```




```
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal:
xlt_number=9 xlt_type=unknown xlt_plan=unknown
Mar 17 01:27:21.004: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal:
number=9725553001 type=unknown plan=unknown numbertype=called
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: No match;
number=9725553001 rule precedence=1
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_match_internal: Matched
with rule 2 in ruleset 1
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: No match;
number=9725553001 rule precedence=1
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_match_internal: Matched
with rule 2 in ruleset 1
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_match: No match;
number=9725553001 rule precedence=1
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/sed_subst: Successful substitution;
pattern=9725553001 matchPattern=9725553 replacePattern=3 replaced pattern=3001
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_subst_num_type: Match Type =
none, Replace Type = none Input Type = unknown
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_subst_num_plan: Match Plan =
none, Replace Plan = none Input Plan = unknown
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal:
xlt_number=3001 xlt_type=unknown xlt_plan=unknown
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal:
number= type=UNKNOWN plan=UNKNOWN numbertype=redirect-called
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_get_RegXrule: Invalid
translation ruleset tag=0
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_match_internal: Error:
ruleset for redirect-called number not found
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_profile_translate_internal: No
match: number= type=UNKNOWN plan=UNKNOWN
Mar 17 01:27:21.008: //-1/xxxxxxxxxxxx/RXRULE/regxrule_vp_translate: calling_number=9
calling_octet=0x0
        called_number=3001 called_octet=0x0
        redirect_number= redirect_type=4294967295 redirect_plan=4294967295
```

The output of the debug is long, but it shows you the profile applied, which rule is applied to the calling party and the result of the translation on the calling party, and which rule is applied to the called party and the result of the translation on the called party.

In a real call, the calling party information would be more complete. In the case of csim start, we can't provide calling party information to the command so the "match" on our voice translation-rule 2 is the last rule which is our catch-all.