Cisco IOS Voice Translations – Part 2: Writing and Applying Voice Translation Rules and Voice Translation Profiles

By Maren Mahoney, Sunset Learning Cisco Unified Communications Specialized Instructor

This is the second 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. This article will cover how those rules are combined to create and apply rulesets and voice translation profiles. Parts 3 and 4 will discuss advanced regular expressions and special applications of voice translation rules, and 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/

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 up to 15 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.

This post will cover the basics of 2, 3 and 4. While that may seem like a lot, the basics flow together nicely. (Part 3 will examine the not-so-basics of 2, 3 and 4.) First let's look at how to create #2, a Voice Translation Rule. A Voice Translation Rule is just a list of individual rules, which you already know how to write.

Here is the scenario: suppose you have a headquarters and two branch offices. All calls for all three sites enter the organization through ISDN PRI circuits at headquarters. As the calls arrive over the circuits, you will need to strip digits from the 10-digit DID numbers and turn them into 4-digit extensions. Since the three locations have very different DID numbers, a single rule will not be able to match/replace the three sets of DIDs. Therefore, we need to make a list of rules.

```
voice translation-rule 1
    rule 1   /7035557…/        /7…/
    rule 2   /9725553[0-4]../  /3.../
    rule 3   /4065559[23]../   /5.../
```

The first rule is for the headquarters and is the same example that we used in Part 1. The second rule is for the first branch in Dallas. Notice that the "[0-4]" in the match string is a variable and can therefore be replaced by a simple "." in the replace string.

The third rule is more interesting. Generally, internal dialplans do not use 9 as the initial digit for a 4-digit extension because it conflicts with using 9 as an outside (PSTN) access code. Since our provider has issued us the 9XXX block within the 555 exchange we can't simply strip the first seven digits to create a 4-digit desk number as we did in rule 1 and rule 2. We need to do something else to resolve the conflict. Several strategies can be used to solve this problem. Our organization has decided to make the desk numbers 5XXX in Montana (our second branch). This means that for rule 3 we need to override the 9 in the match string with a 5 in the replace string. (Another strategy would be to use 8, rather than 9, as the PSTN access code.)

We now have a set of rules that we can apply to the called-party of inbound calls.

What if, for those same inbound calls, we want to modify the caller-id to prefix the access numbers (9, 91 or 9011) so that the calls can be dialed easily from a user's call history? To accomplish this, we need another set of rules:

```
voice translation-rule 2
    rule 1   /^…….$/                /9&/
    rule 2   /^[2-9]..[2-9]……$/     /91&/
    rule 3   /^1[2-9]..[2-9]……$/    /9&/
    rule 4   /.*/                   /9011&/
```

The ampersand (&) in the replace string means "put everything from the match string here". So, in this voice translation rule, a 7-digit caller-id would be prefixed by a 9, a 10-digit number that conformed to the North American Numbering Plan would be prefixed by 91 (or just 9, if the 1 was already in the caller-id) and all other calls would, presumably, be international and would be prefixed by 9011.*

(Note: This method of manipulating caller-id is clumsy and definitely not a best-practice. I use it here as a simple example for the purpose of explaining Voice Translation Rules and Profiles. In Part 3 of this series, we will look at a better way of manipulating caller-id.)

Now we have a pair of Voice Translation Rules that need to be applied to calls inbound over our ISDN circuits. The next step is to combine the two Voice Translation Rules into a Voice Translation Profile. When you create a Voice Translation Profile, you have to give it a "name" so that it can be referenced by the dial-peer or port. You also have to declare which ruleset should apply to called-party and which ruleset should be applied to caller-id. Here is our example:

```
voice translation-profile PSTN-IN
    translate called 1
    translate calling 2
```

* The match string /.*/ assumes that there is at least one digit in the string, but after that the string can be any length. This is different than a match string of // which would match a NULL field (no digits).

The name used to identify a voice translation-profile can be alphanumeric, and can be quite long. Feel free to create a friendly name that helps you remember what a profile is for when you – or someone else – looks at it months after you created it. For example, you might call a rule "in-from-MT-internal-to-Intl" to identify calls inbound from internal extensions at your Montana location where the dialed number is international.

So far we have created rules, put the rules into Voice Translation Rule rulesets, and assembled rulesets into a Voice Translation Profile. All that remains is to apply the Voice Translation Profile to a port or dial-peer.

In our example we have an ISDN PRI on, let's say, port 0/0/0. To apply the Voice Translation Profile to the port, we need to define which profile to use and whether the profile should be applied to incoming or outgoing calls.

```
voice-port 0/0/0:23
    translation-profile incoming PSTN-IN
```

When applying a Voice Translation Profile, it is important to note that the "name" of the profile is case-sensitive – and the router will not warn you if you make a mistake. For instance, what if you type the name of the profile in as "PSTN-IN" but apply it with "pstn-in"? In that case, the router will take the command as typed, and when a call comes in it would look for a profile called "pstn-in". Not finding one, it would not apply any ruleset at all and pass the digits as-received. (In Part 4 we will look at a debug that will help identify this type of problem.)

If you have more than one ISDN circuit, you have two choices: either apply the profile to each incoming voice port one-by-one, or define a default incoming POTS dial-peer and apply the profile to that. (Having default incoming dial-peers for POTS and VOIP calls is a recommended practice.) If you have multiple incoming dial-peers, you would apply incoming translation profiles as needed.

```
dial-peer voice 1 pots
        incoming called-number .
        translation-profile incoming PSTN-IN
        direct-inward-dial
```

That could be read as: For all calls coming in on any POTS circuit, as long as there is at least one digit in the called party, then apply translation-profile PSTN-IN and immediately attempt to match an outbound dial-peer.

For inbound calls we are done – but what about outbound calls? Another action your router can take is to properly format your internal users' caller-id for calls outbound to the PSTN. We created rules like this in Part 1. Here is a ruleset for our three sites:

```
voice translation-rule 3
    rule 1   /^7…$/        /7035557…/
    rule 2   /^3[0-4]..$/   /9725553.../
    rule 3   /^5[23]..$/    /4065559.../
```

And here is the voice translation-profile that has the router use voice translation-rule 3 to modify caller-id:

```
voice translation-profile PSTN-OUT
    translate calling 3
```

And finally, here is the voice translation-profile applied outbound on voice-port 0/0/0:23 (Note that the translation profile for incoming calls is still also applied to the port):

```
voice-port 0/0/0:23
    translation-profile incoming PSTN-IN
    translation-profile outgoing PSTN-OUT
```

Or, you could apply the translation-profile to dial-peers used for PSTN calls:

```
dial-peer voice 9 pots
    destination-pattern 9[2-9]..[2-9]…...
    translation-profile outgoing PSTN-OUT
    port 0/0/0:23
```

Those are the basic building blocks of implementing digit manipulation on an IOS router. It can be as simple or as complex as you need it to be. A good suggestion for helping to build the pieces is to write each step out in words, and then figure out the exact syntax for each step. This is similar to writing an outline before you write a business report, or similar to building what programmers call “pseudocode” before starting to build the actual application.

In the next post we will look at some of the more complex arguments of, and applications for, voice translations such as identifying calls based on type of number (TON) or blocking calls based on caller-id.