How Action Sets Work

Introduction

The purpose for a SBC is to route calls – matching and modifying telephone numbers, adding diversions, applying calling names. Call Routes and Transformation Tables are the tools typically selected to perform these functions, but they are not the only, nor the most powerful, manipulation tools available in the SBC.

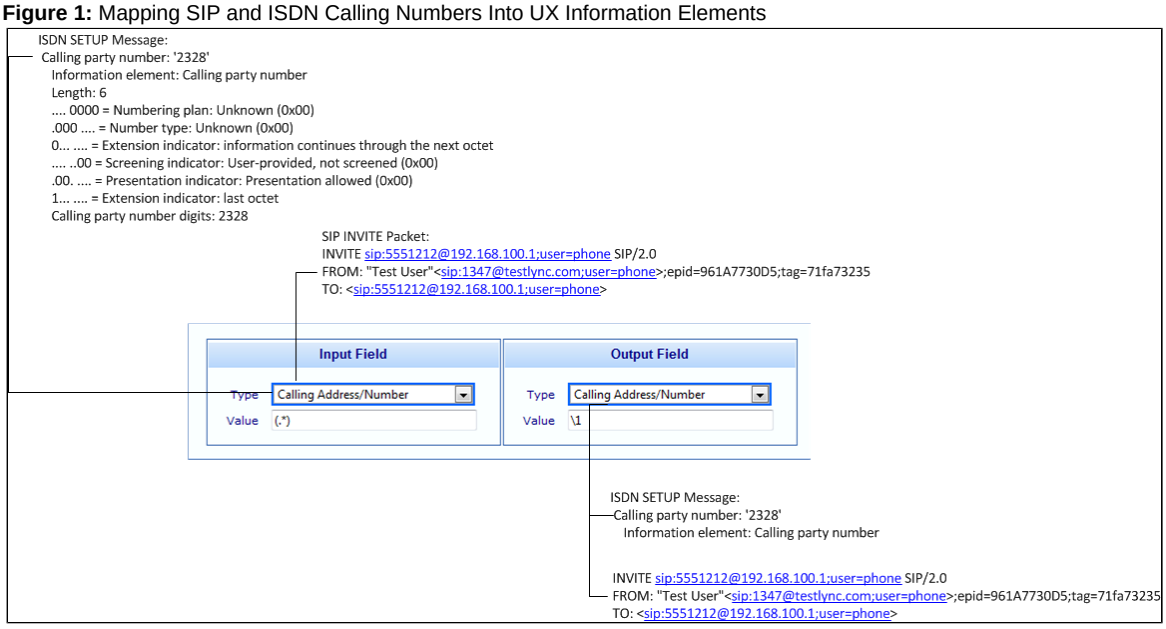
Action Sets operate similarly to Call Routes, but because Action Sets are performed before call routing, Action Sets provide the capability to perform a high degree of number normalization before engaging the call router. Even more importantly, Action Sets incorporate a powerful set of programmatic functionality that permits the construction of calling applications.

In this article you will learn the basics of Action Sets and their capabilities beginning with a quick review of basic call routing, look at the Action Set components, reveal the steps to create an Action Set, and finally follow the Action Set processing step-by-step.

Basic SBC Call Routing

Stored Information Elements (IEs)

Figure 1: Mapping SIP and ISDN Calling Numbers Into UX Information Elements



When a call arrives at SBC, the information contained in the incoming message (called number, calling number, name, etc.) is stored into variables within SBC. These variables are called Information Elements, a nomenclature lifted from ISDN definitions. So, for example, when a SIP call arrives, the number in the FROM header is stored in SBC's Calling Number IE. All arriving SIP headers or ISDN messages are similarly stored.

Transformation Tables

With the IEs stored, we need a way to compare and manipulate those IEs as a function of routing calls. Within the SBC, Transformation Tables provide this functionality. For instance, in a Tranformation Table, called numbers can be matched and changed, caller's names can be looked up in AD and recorded in the Calling Name IE, or diversion headers can be added. This Transformation Table functionality works the same regardless of whether it is called from a Call Route or Action Set. However, there is a difference in Actions Sets and the Call Router when it comes to the **persistance** of manipulations. We'll talk about that in a second.

The Call Router

With basic call routing, Transformation Tables are tied to Call Routes. As a call route is executed, the associated Transformation Table is used to match, and possibly, manipulate the data within the IEs. For example, a Call Route to Lync can be associated with a Transformation Table that checks the called number IE against Active Directory's msRTCSIP-Line attribute. If a match is found, the call is routed to Lync. **If the Transformation Table fails, all the IEs are restored to their original values before the next Call Route Entry is executed.**

Call Routes (group into Call Route Tables) are executed in order from top to bottom. This processing path is immutable--every entry in a call Route Table is processed one at a time from top to bottom (unless the call route entry is disabled).

This fixed top-to-bottom processing works perfectly well for the vast majority of circumstances. But what if we wanted to do further processing based upon success or failure of the msRTCSIP lookup? Maybe the circumstances are such that we want to send the call to a different Call Route Table if the msRTCSIP succeeds, possibly to a shared Transformation Table that performs all the static transformations. This can't be done from a Call Route. From a Call Route, when the Transformation Table succeeds, the call is automatically routed.

In order to perform additional processing, we need a function that permits *programmatic* call processing. Action Sets are the answer. They perform a similar function as the Call Route Table, but with a completely different approach to processing.

SBC Action Sets

Action Sets are called from the ingress Signaling Group and bypass the Call Route Table setting (if one is configured in the SG). Action Sets provide the opportunity to process transformations **before** sending the call to the call router. Think of it as a chance to normalize all the IEs before routing a call.

Not only do Action Sets provide normalization, but they also provide **two** levels of if/then logic: *Actions* and *Execute If*. These two features are the foundation for creating SBC calling application solutions.

Actions

Actions are configured from the *Action Configuration* menu. The available Actions are:

Action	Usage
Route Call	Route a call to the specific call route table
Send Alert	Send Alert/Ring. To be implemented in the future with the One Number Fax function
Send Connect	Send Connect/200 OK. To be implemented in the future with the One Number Fax function
Route Call, Await Connect Timer	Routes the call and starts a timer. Automatically continues execution of the Action Set if the call is not answered before the timer expires. Can be used to create applications to send unanswered calls to a local voicemail store.
Generic Timer	Provides a delay function in processing entries in the Action Set
Invoke Action Set	Invoke a different Action Set. Provides the ability to continue execution with a different Action Set.

Using these Actions, we can do things like:

- Route a call using a specific call route table
- Disconnect a call if a transformation table fails
- Continue processing a call using a different Action Set
- Set an Action to occur if a routed call Rings with No Answer

Once some Actions are created, you need a way to use them. That's where *Action Sets* come into play.

Action Set Table

An Action Set Table contains the list of entries that are executed when that particular Action Set is called (from the ingress SG). Unlike a Call Route Table, the entries in an Action Set are not necessarily processed in sequential order. Instead, by using *Action On Success*, *Action on Failure*, or *Execute If* (in any combination), you can control which Action Set entries are processed.

Figure 2: Action Set With Various Actions and Execute Ifs

Execute If	Transformation Table	Action on Success	Action on Failure
Always	Normalize Incoming Numbers	Continue	Failure - Disconnect Call
Always	Get Calling Name - concat	Continue	Continue
Always	Is the call to EUM SA?	Continue	Continue
Prior Success	AutoID Caller to EUM SA	Action, Route to EUM	Continue
Prior Failure	Is it a FAX number?	Action, Route to EUM	Continue
Always	Is the call to a LYNC Analog device?	Action, RNA Timer, Route to Lync	Continue
Prior Timeout	EUM Leave voicemail, get callee's umbox number	Action, Route to EUM	Failure - Disconnect Call
Always	Is the call to a Lync device?	Action, Route to Lync	Continue
Always	Restore Original Calling Number	Action. Route to PBX	Failure - Disconnect Call

Actions within Action Sets

When an Action Set Entry is executed, the associated Transformation Table is evaluated to be either TRUE or FALSE (the result of whether the Transformation Table succeeded or failed in its matching). So far, exactly the same as an entry in a Call Route Table.

One difference between a Call Route and Action Set is that each entry in a Action Set contains an *Action on Success* and *Action on Failure* setting. These two settings provide control over the next step in processing based upon the outcome of the associated Transformation Set:

- If the Transformation Set succeeds, the Action configured in the *Action on Success* is performed.
- If the Transformation Set fails, the Action configured in the *Action on Failure* is performed.

Figure 3: Actions in Action Sets

Transformation Table	Action on Success	Action on Failure
Normalize Incoming Numbers	Continue	Failure - Disconnect Call

In the Action Set Entry above, processing will *Continue* with the next entry (if any) if the transformation is TRUE; but, it will disconnect the call if the transformation is FALSE.

So, unlike the Call Routes, we can control what happens when a Transformation Table succeeds or fails.

Using Actions, you can control how the processing proceeds: Route the call; disconnect the call; continue with the current or a different Action Set, etc.

Execute If:

When an Action Set is executed, the execution of individual Action Set entries is controlled by the *Execute If*. Whereas Call Route entries can only be Enabled or Disabled, the execution of Action Set entries can be controlled **based on success or failure of the previous entry**.

Looking at the configuration below, we can see that the first three Action Set entries are *Always* executed. You can also see that the Actions for the third entry's *Success* and *Failure* are configured to *Continue*. Regardless of the outcome of the third entry's Transformation Table, processing will continue. The question is with which entry?

Figure 4: Execute If

Execute If	Transformation Table	Action on Success	Action on Failure
Always	Normalize Incoming Numbers	Continue	Failure - Disconnect Call
Always	Get Calling Name - concat	Continue	Continue
Always	Is the call to EUM SA?	Continue	Continue
Prior Success	AutoID Caller to EUM SA	Action, Route to EUM	Continue
Prior Failure	Is it a FAX number?	Action, Route to EUM	Continue

Notice that the fourth entry's *Execute If* is configured to *Prior Success*. This means that the fourth entry will run *only if* the third entry is successful. Alternatively, if the third entry is *not* successful, the fourth entry will be skipped and the fifth entry will run.

You can see that using *Execute If* the order of entry execution can be controlled.

- If an Action Set entry succeeds, the next entry that processes will be set for either *Always* or *Prior Success*.
- If an Action Set entry fails, the next entry that processes will be set for either *Always* or *Prior Failure*.

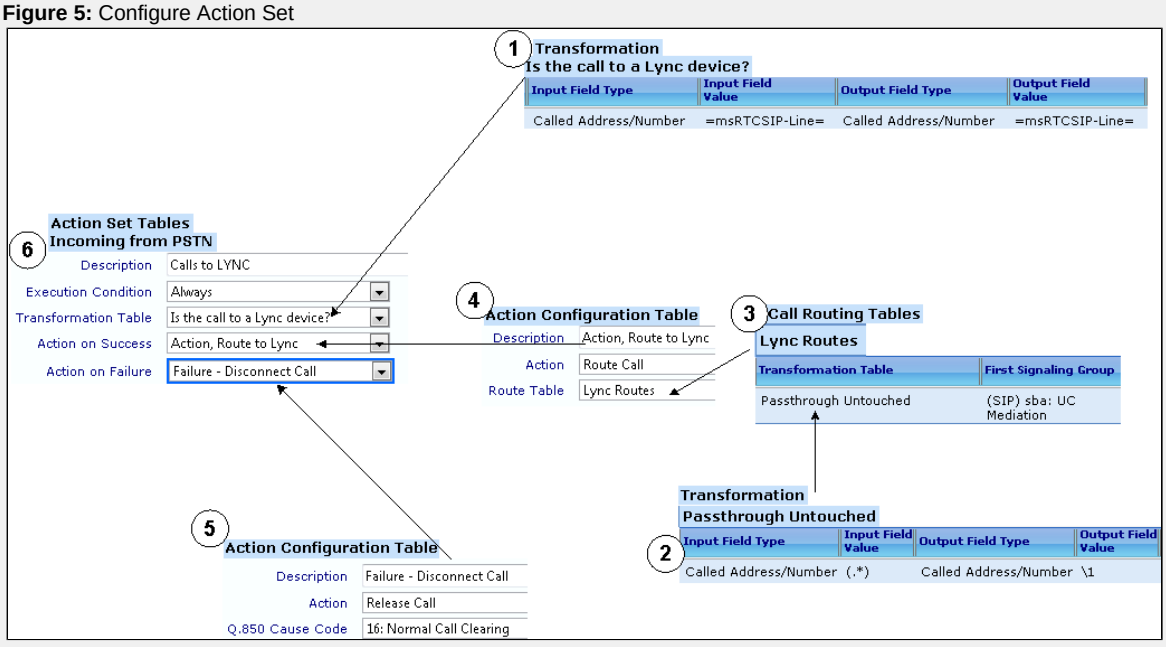
With *Execute If*, you can control which Action Set entries are executed based upon the success or failure of the previous entry.

Unlike Call Route entries, the IEs are **not** restored for each different Action Set Entry. Each change to an IE (in a successful Transformation Table) is carried **throughout the entire Action Set Table**. In this way, the IEs can be normalized over several entries before the call is routed. If you wish to retain a particular value, place it into one of the user-defined IEs (UserValue1-UserValue5).

Configuring an Action Set

Much like the configuration of Call Routes, there are a number of inter-dependecies in configuring Action Sets. The diagram (below) will help guide you through the process.

Figure 5: Configure Action Set

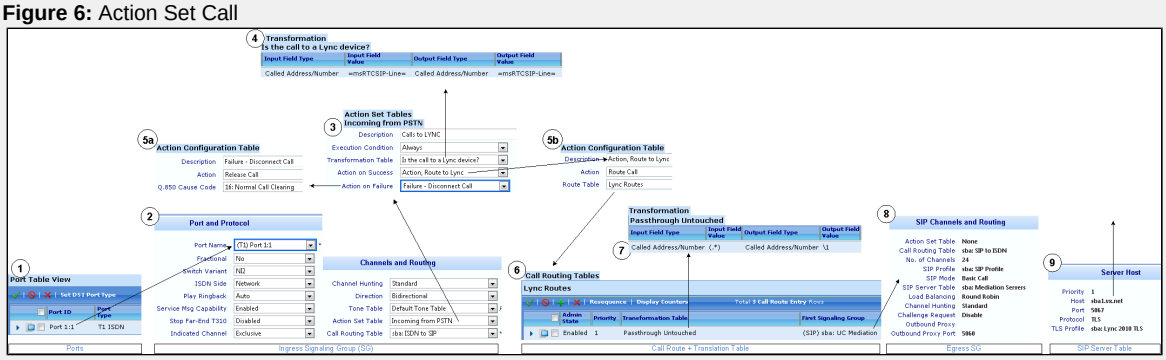


1. Create a Transformation Table to be used by the Action Set
2. Create a Transformation Table to be used by the Call Route.
3. Create a Call Route Table and add any Call Route entries required to complete the calls.
4 and 5. Add any desired Actions in the *Action Configuration* table.
6. Add an Action Set table and populate it with whatever entries are required to complete the desired IE processing.

How Action Sets Process a Call

Let's put it all together and see what an Action Set call looks like.

Figure 6: Action Set Call



1. Call arrives at ingress port
2. Ingress Signaling Group sends the call to the configured Action Set. Note that the Call Routing Table configuration is ignored when an Action Set Table is defined.
3. The first entry in the Action Set Table executes and
4. Processes the associated Transformation Table.
5. If the Transformation Table fails, the call is disconnected per the *Action on Failure* configuration. If the transformation succeeds, the call is sent to the *Lync Routes* Call Route Table.
6. The *Lync Routes* Call Route Table executes and
7. Processes the associated translation table.
8. Finally, the call is sent to the egress Signaling Group and
9. To the destination SIP Server.

Next Steps

SBC AD Integrated Application Solutions using Action Sets details a real-world application of Actions Sets to:

- Normalize Numbers
- Blacklisted Numbers
- Find a Calling Name
- Send calls to Exchange for voicemail retrieval
- Calls to Fax Numbers
- Permit voicemail for analog Lync devices
- Send calls to Lync Users

- Use the PBX as the route of last resort

This application employs the Action Set's full capabilities. Reviewing this document should help solidify your understanding of Action Sets and put you on the path to creating your own.