OBIHAI
technology, inc.

Obihai Technology, Inc.

OBi 500 Series –

SIP Trunking Guide

Models:

OBi508 – 8-Port Voice Service Bridge and Terminal Adapter

OBi504 – 4-Port Voice Service Bridge and Terminal Adapter

January 2016

SIP Trunking

There are 9 ITSP Profiles and 9 SP Services available in each OBi500 device. Each SP Service can be configurted as a SIP Trunk in the following sense:

1. Single REGISTRATION per SP account
2. Multiple DID numbers for incoming calls and outgoint calls
3. Incoming call routing based on called DID numbers, to ring any combinations of phone ports simultaneously or a hunt group
4. Static Call Forwarding according to DID numbers via hunt groups
5. Outgoing caller identity based on the phone port that originates the call
6. Independent Call Fowarding (302 redirect) per phone port (TBD)?
7. MWI Status NOTIFY routing/handling based on DID numbers (TBD)?
8. Option to insert user=phone for caller identity (for a DID number) in outgoing calls

Configuration of a SIP Trunk on SP1 (ITSP Profile A) on OBi508

In this example, we use SP1 on ITSP Profile A as SIP trunk.

Use Case:

- All 8 phone ports use SP1 for outgoing calls
- Each phone port has a DID number, +14089991001, +14089991002, … +14089991008, respectively
- Let *mytrunkid* be the Pilot UserID of the SIP trunk

Configuration:

Parameter	Value	Description
ITSP Profile A–SIP:: **X_SpoofCallerID**	`true`	Enable this option to let the device insert a UserID in the FROM header of outbound INVITE that is different from the SIP Account UserID
ITSP Profile A–SIP:: **X_UserEqPhone**	`true`	Enable this option to let the device insert a ;user=phone (URI) parameter in the Request-URI and TO header of outbound INVITE (for all calls on SP1). Do not enable this option if the server does not expect to see this URI parameter
SP 1 Service:: **AuthUserName**	`mytrunkid`	Configure the SIP Account credentials. If the UserID for authentication is different from the SIP UserID, the SIP UserID must be configured in the **URI** parameter and authentication UserID in the **AuthUserName** parameter.
SP1 Service:: **AuthPassword**	The authentication password corresponds to mytrunkid	
SP1 Service:: **URI**		

SP1 Service:: **X_InboundCallRoute**	`{>(@.4089991001):ph1},`
	`{>(@.4089991002):ph2},…,`
	`{>(@.4089991008):ph8}`
		Configure incoming calls to a specific DID number to ring the corresponding phone port only. The @. syntax matches any number of characters before the last 10 digits of the called number (for

		example: + or +1 or +001). Note that you can also ring any combination of the phone ports for a given called number, or ring a hunt group (GP) defined in the device configuration
Phone 1 Port:: **OutboundCallRoute**	`{(Mpli):pli(+14089991001;context=abd.com;;user=phone>$2)}`	Calls made from each phone port to assume a different identify based on assigned DID number. This identify is shown as the UserID in the From header (and Remote-Party-ID header if enabled) of the outbound INVITE. The Contact header of the INVITE on the other hand uses the Pilot UserID (the SP Account USERID) as the UserID. `$2` represents the target number that the user dials. Note that additional call routing rules may be added for other call destinations. Note that UserID parameters after the first `;` in the UserID are included as is as part of the user id, but parameters after the first `;;` are inserted as URI parameters.
Phone 2 Port:: **OutboundCallRoute**	`{(Mpli):pli(+14089991002>$2)}`	
…	`{(Mpli):pli(+1408999100`*n*`>$2)}` where *n* = 1, 2, …, 8	
Phone 8 Port:: **OutboundCallRoute**	`{(Mpli):pli(+14089991008>$2)}`	
Phone 1 Port:: **PrimaryLine**	`SP1 Service`	All phone ports to use SP1 as the primary line (pli).
Phone 2 Port:: **PrimaryLine**	`SP1 Service`	
…	`SP1 Service`	
Phone 8 Port:: **PrimaryLine**	`SP1 Service`	

Notes:

- You can tell the device to insert a `user=phone` URI parameter in the FROM header of outbound INVITE (and Remote-Party-ID header if enabled) when spoofing the calling UserID with a DID number, by including a `;;user=phone` parameter after the spoofed userid in an OutboundCallRoute rule. In fact, everything in the rule's argument after the first `;;` is included as URI parameter(s).

Call Forward Settings Per Phone Port

There is a set of call forward settings per phone port, as shown in the following table:

Parameter **Phone Port *n* –** **Call Forwarding::**	**Description**
ForwardAll	Enable forwarding all calls unconditionally. Value must be `true` or `false`.

ForwardAllNumber	The number to forward all calls to unconditionally. It may include the Caller-ID number to use as the UserID of the SIP/Diversion header. Examples:
	`SP1(1000)`
	`SP1(+14089991003>1000)`
	`SP1(+14089991003;;user=phone>1000)`
	`SP1(+14089991003;context=test.com;;user=phone>1000)`

ForwardOnBusy	Enable forwarding calls when the phone port is busy. Value must be `true` or `false`.

ForwardOnBusyNumber	The number to forward calls to when the phone port is busy. See **ForwardAllNumber** for examples.
ForwardOnNoAnswer	Enable forwarding calls when the call is not answered after a few rings. Value must be `true` or `false`.
ForwardOnNoAnswerNumber	The number to forward calls to when the call is not answered after a few rings. See **ForwardAllNumber** for examples.
ForwardOnNoAnswerRingCount	The number of rings to wait before triggering call forwarding on no answer, where one ring is equivalent to six seconds

Note that call forwarding from a phone port is implemented only by ***bridging*** two calls together by the device. That is, the device maintains the call with the original incoming caller (in the ringing state), while making a new call to the call forward target number. When the target rings or answers (whichever happens first), it bridges the two call legs together. The SIP/INVITE to the call forward target will have the original caller's Caller ID in the FROM header, and the Request-UserID of the original SIP/INVITE in a **DIVERSION** header. If the original Request-UserID has a `user=phone` parameter, the DIVERSION header will carry the same URI parameter as well.

DND Setting Per Phone Port

There is a DND (Do Not Disturb) setting per phone port, as shown in the following table:

Parameter **Phone Port *n* –** **Calling Features::**	**Description**
DoNotDisturbEnable	Enable Do Not Disturb on a phone port. Value must be `true` or `false`.

Using Star Codes to Change Per-Phone-Port Call Forward and DND Settings

Assuming a different StarCodeProfile is used for each phone port, one can define local star codes to enable/disable the call forward settings per phone port, as shown below:

For Phone Port n (n = 1, 2, 3, …, 8) using Star Code Profile x (x = A, B, …. H)

```
*72, Cfwd All, coll(PHn($Cfan)), set(PHn($Cfa),1)
*73, Disable Cfwd All, set(PHn($Cfa), 0)
*60, Cfwd Busy, coll(PHn($Cfbn)), set(PHn($Cfb),1)
*61, Disable Cfwd Busy, set(PHn($Cfb), 0)
*62, Cfwd No Ans, coll(PHn($Cfnn)), set(PHn($Cfn),1)
*63, Disable Cfwd No Ans, set(PHn($Cfn),0)
*78, Enable DND, set(PHn($Dnd),1)
*79, Disable DND, set(PHn($Dnd),0)
```

It is also possible to use the same star code profile to change the values of call forwarding and DND settings for each phone port, with the generic phone port versions of corresponding star code variabes; the phone port whose settings to be changed in this case is the one that the star code is dialed from:

```
*72, Cfwd All, coll($PhCfan), set($PhCfa,1)
*73, Disable Cfwd All, set($PhCfa, 0)
*60, Cfwd Busy, coll($PhCfbn), set($PhCfb,1)
*61, Disable Cfwd Busy, set($PhCfb, 0)
*62, Cfwd No Ans, coll($PhCfnn), set($PhCfn,1)
*63, Disable Cfwd No Ans, set($PhCfn,0)
*78, Enable DND, set($PhDnd,1)
*79, Disable DND, set($PhDnd,0)
```

MWI Notfication Routing

Your OBi device can handle SIP/NOTIFY for MWI status based on the DID number in the Request-UserID of the NOTIFY request. The same processing is used inside or outside the context of a Subscription dialog. You can define routing rules to process the MWI status by any combination of phone ports. There are two settings for this under each SP*n* Service: **X_MWIRoute** for basic MWI handling (by plaing Stutter Tone) and **X_VMWIRoute** for VMWI. For example:

```
{(@.4089991001)>ph1},{(@.4089991002)>ph2},…,{(@.4089991008)>ph8},{ph7,ph8}
```

In the last example, SIP/NOTIFY sent for a particular DID number is handled by a corresponding phone port only, except the last rule which is a catch-all rule for SIP/NOTIFY sent for anything else.

Hunt Groups

Hunt Groups are configured on the device web page with the same name or under the parameter object **VoiceService.1.X_HuntGroups.** in a configuration file (via device provisioning that is). There are 10 parameters **Goup1** – **Group10** in this object, where each a hunt group can be specified as described below.

The general syntax for a hunt group parameter value is:

GP(*name*) = {*ringlist*};*par*;*par*;…;*par*

where *name* is string to uniquely identify the hunt group within the device and to refer to the hunt group in call routing rules, *ringlist* is a list of destinations to alert for incoming calls routedd to the group, and each *par* is a valid parameter=value pair that specifies additional behavior. Extra white spaces are allowed. All syntaxes are *case-insensitive*.

Some examples:

```
GP(1)={ph1+ph2+ph3+ph4}
```
Synopsis: Try to ring ph1, ph2, ph3, ph4 sequentially. The hunt stops at the first phone port that rings (that is not busy). The device replies busy if all 4 phone ports are busy. You may want to disable CallWaiting on all the ports if you do not want a phone port to ring when it is already on a call

```
GP(2)={ph1+ph2+ph3+ph4};dur=20;np=0;na=sp1(+14089991001>+15103123344)
```
Synopsis: Similar to the last example but ring the group for up to 20s only with no repetition. If no answer, call +15103123344 on SP1 with the DID number +14089991001 as calling UserID, and when it rings, bridge this call with the caller

```
GP(99)={ph;du=20+ph2+sp1(+16509991234),sp1(+14089991234)};np=1;alg=SEQ
```

Synopsis: Ring ph1 for up to 20s, then ph2. If ph2 is busy, call +16509991234 and +14089991234 on SP1 simultanrously with the pilot userid on SP1 as the calling identity

```
GP(sales)={ph1+ph2+ph3+ph4;du=30;rg=2;np=2;alg=RR;na=gp(all)
```
Synopsis: Pick one of ph1, ph2, ph3, and ph4 randomly to start ringing. If busy or no answer for 30s, ring the next phone port in the list in a round robin (alg=RR) fashion and repeat the list twice (np=2). If no answer, ring the group GP(all)

```
GP(all)={ph1,ph2,ph3,ph4,ph5,ph6,ph7,ph8};du=20
```
Synopsis: Ring all 8 phone ports simultaneously for up to 20s

Ringlist:

Each member in a ringlist is referred to as a terminal. A terminal can be a phone port or a number to call. Be cautious that you do not refer to another GP implicitly or explicitly.

Parameters:

du: Duration in seconds

```
// - du=* => inifinite duration
```

alg: Algorithm to use

```
// - alg=algorithm to ring the phones, seqential, round-robin, etc.
```

np: Number of passes

```
// - np=0 => only ring 1 phone, and triggers na when ring duration is up
```

rg: Ring type (distinctive ring)

```
// - rg=ring type (1-10, one of the 10 available ring patterns in a ring rofile
```

na: Where to redirect the call when no one answers

```
// - unspecified duration => ring forever (i.e.,try next only if busy)
// - no recusive gp definition
// - na=who to call if not available
// - all syntax case-insensitive
```