TECHNICAL REPORT

TR-104
Provisioning Parameters for VoIP CPE

Issue: 2
Issue Date: March 2014

Notice

The Broadband Forum is a non-profit corporation organized to create guidelines for broadband network system development and deployment. This Broadband Forum Technical Report has been approved by members of the Forum. This Broadband Forum Technical Report is not binding on the Broadband Forum, any of its members, or any developer or service provider. This Broadband Forum Technical Report is subject to change, but only with approval of members of the Forum. This Technical Report is copyrighted by the Broadband Forum, and all rights are reserved. Portions of this Technical Report may be copyrighted by Broadband Forum members.

THIS SPECIFICATION IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NONINFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER the Forum, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS SPECIFICATION.

Broadband Forum Technical Reports may be copied, downloaded, stored on a server or otherwise re-distributed in their entirety only, and may not be modified without the advance written permission of the Broadband Forum.

The text of this notice must be included in all copies of this Broadband Forum Technical Report.

Issue History

Issue Number	Approval Date	Publication Date	Issue Editor	Changes
1	September 2005		Jeff Bernstein, 2Wire Barbara Stark, BellSouth	Original. Defines VoiceService:1.0.
2	3 March 2014	24 April 2014	Alexandre François, Orange Grégory Foucher, Orange	Original. Defines VoiceService:2.0.

Comments or questions about this Broadband Forum Technical Report should be directed to help@broadband-forum.org.

Editor	Alexandre François Grégory Foucher	Orange Orange
BroadbandHome™ WG Chairs	Jason Walls John Blackford	QA Café Pace

Table of Contents

List of Figures

List of Tables

Executive Summary

TR-104 defines version 2 of the TR-104 [7] VoiceService data model (VoiceService:2). The VoiceService:2 data model applies to all types of TR-069-enabled devices, including End Devices, Residential Gateways, and other Network Infrastructure Devices.

The evolution to VoiceService:2 was necessary in order to resolve some fundamental limitations in the VoiceService:1 data model, which only covers VoIP gateways with FXS ports. The VoiceService:2 data model defined in this Technical Report has been remodeled and now covers non-VoIP telephone network connections, local communication possibilities and advanced telephony features such as can be found in most PBX.

Updates for Issue 2 include:

- Support for FXO, DECT, ISDN.
- Support for SIP Proxy and Registrar.
- PBX functionalities and advanced telephony features.

1 Purpose and Scope

1.1 Purpose

This specification defines the data model for provisioning CPE with voice services by an Auto-Configuration Server (ACS) using the mechanism defined in TR-069 [2].

1.2 Scope

The goals of this specification are as follows:

- Accommodate VoIP devices that support multiple distinct VoIP services, each potentially with multiple distinct lines.
- Support the use of both SIP [4] and MGCP [5] signaling protocols.
- Support various types of VoIP CPE including VoIP endpoints, SIP outbound proxies, and SIP back-to-back user agents.
- Support a PBX embedded in the CPE.
- Support DECT handsets associated with one of the CPE device's DECT bases.
- Support a VoIP Gateway (for example SIP <-> ISDN)

Figure 1 illustrates the top-level VoiceService:2 data model structure. Figure 2, Figure 3, Figure 4 and Figure 5 illustrate the data model structure in greater detail. See Section 5 for the complete list of objects.

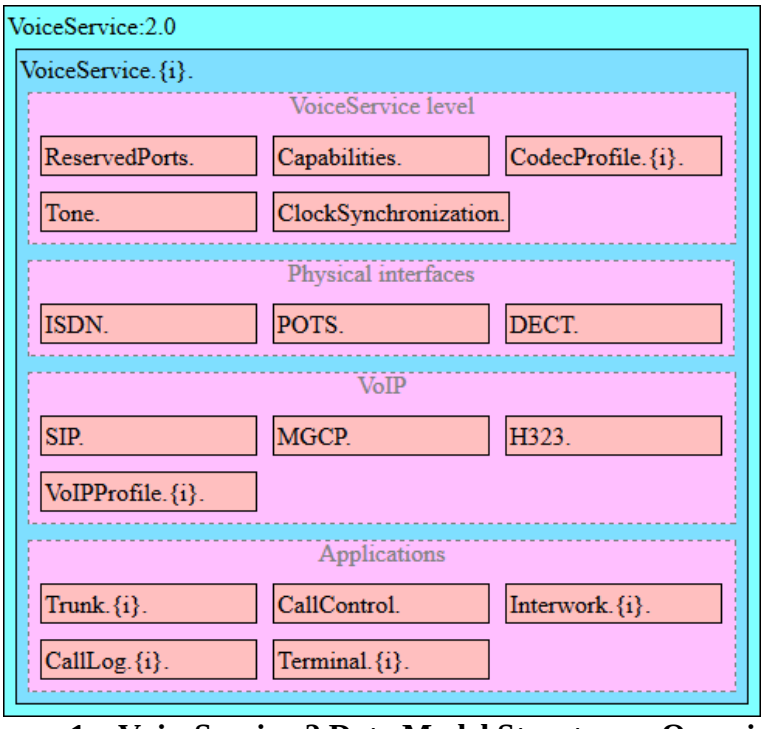

Figure 1 – VoiceService:2 Data Model Structure – Overview

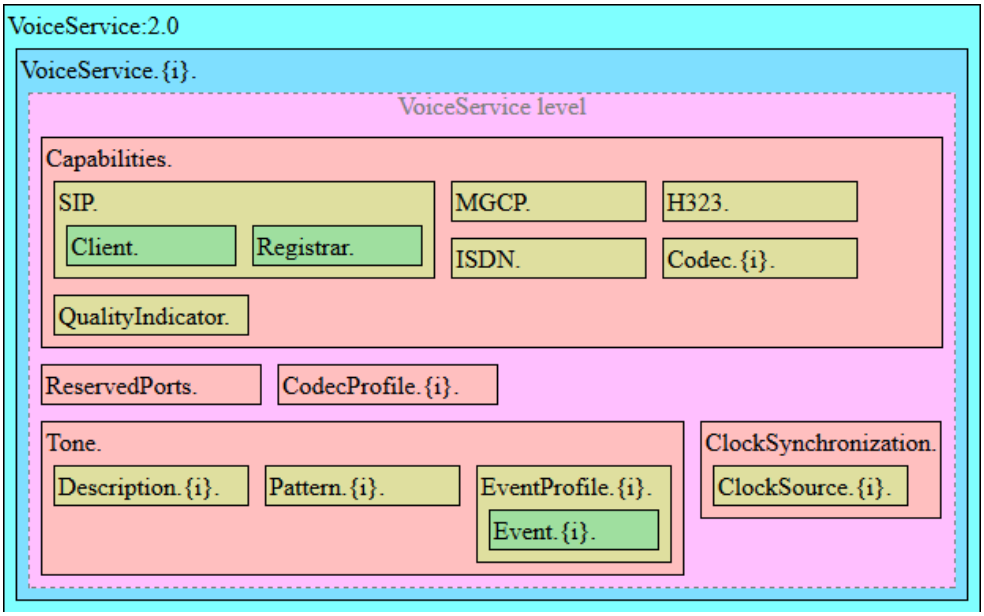

Figure 2 – VoiceService:2 Data Model Structure – VoiceService Level

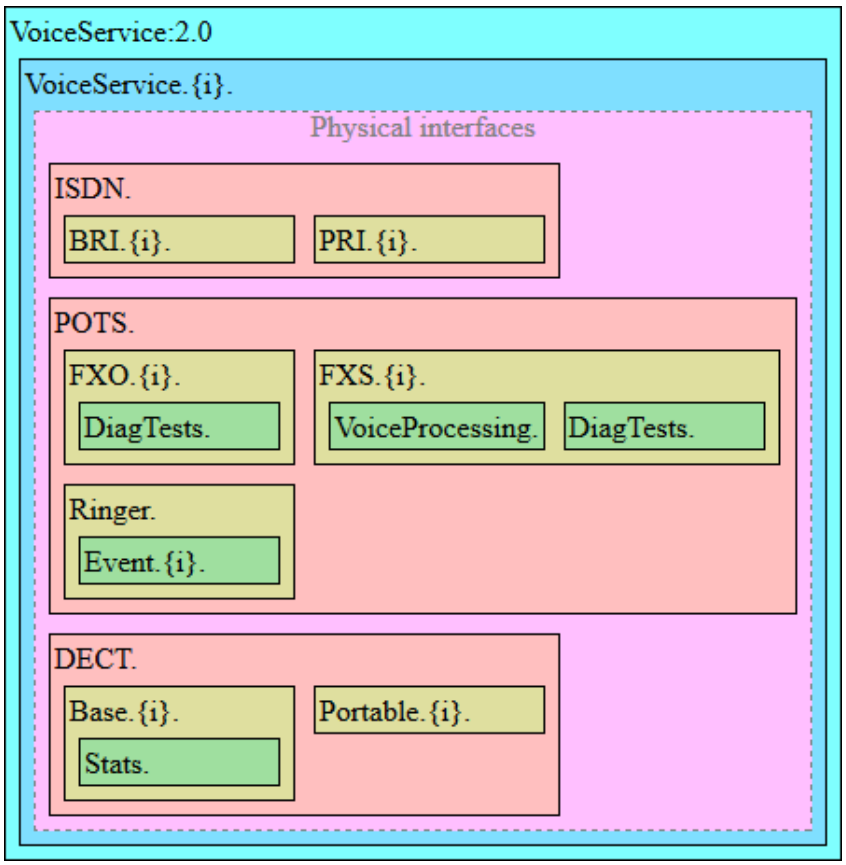

Figure 3 – VoiceService:2 Data Model Structure – Physical Interfaces

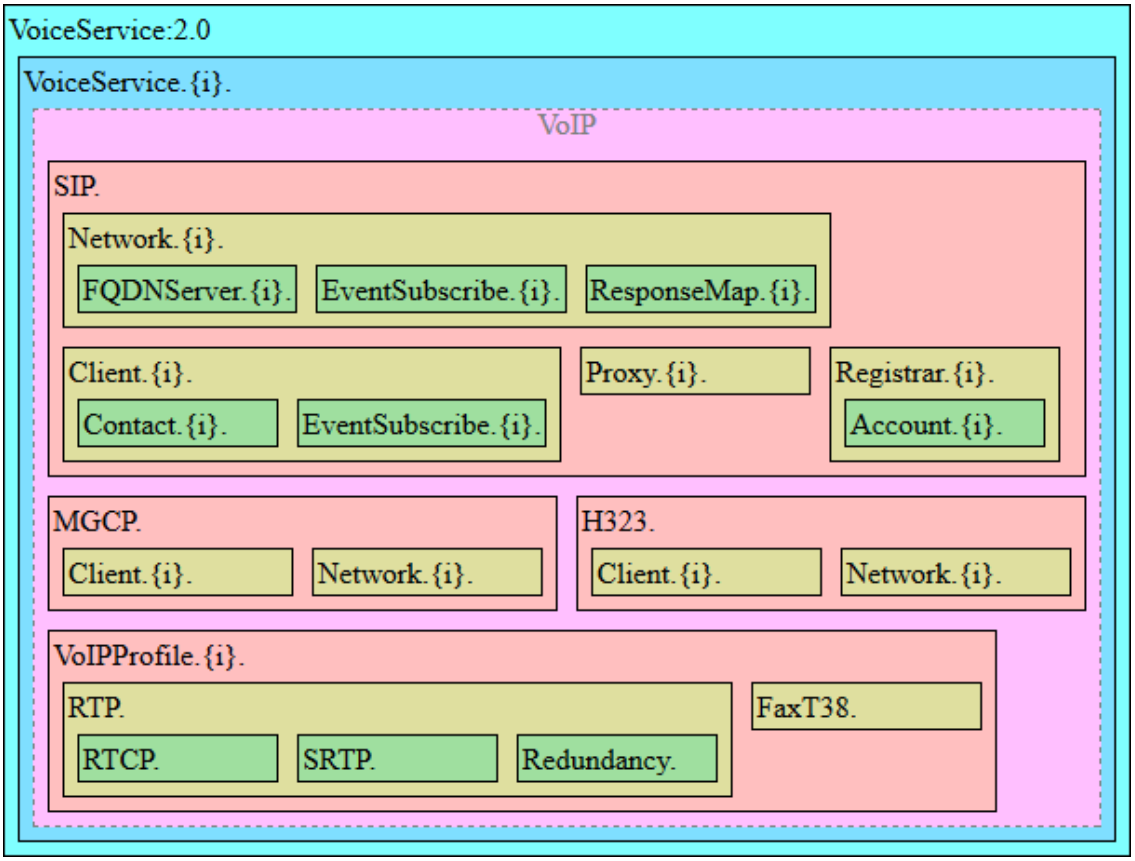

Figure 4 – VoiceService:2 Data Model Structure – VoIP

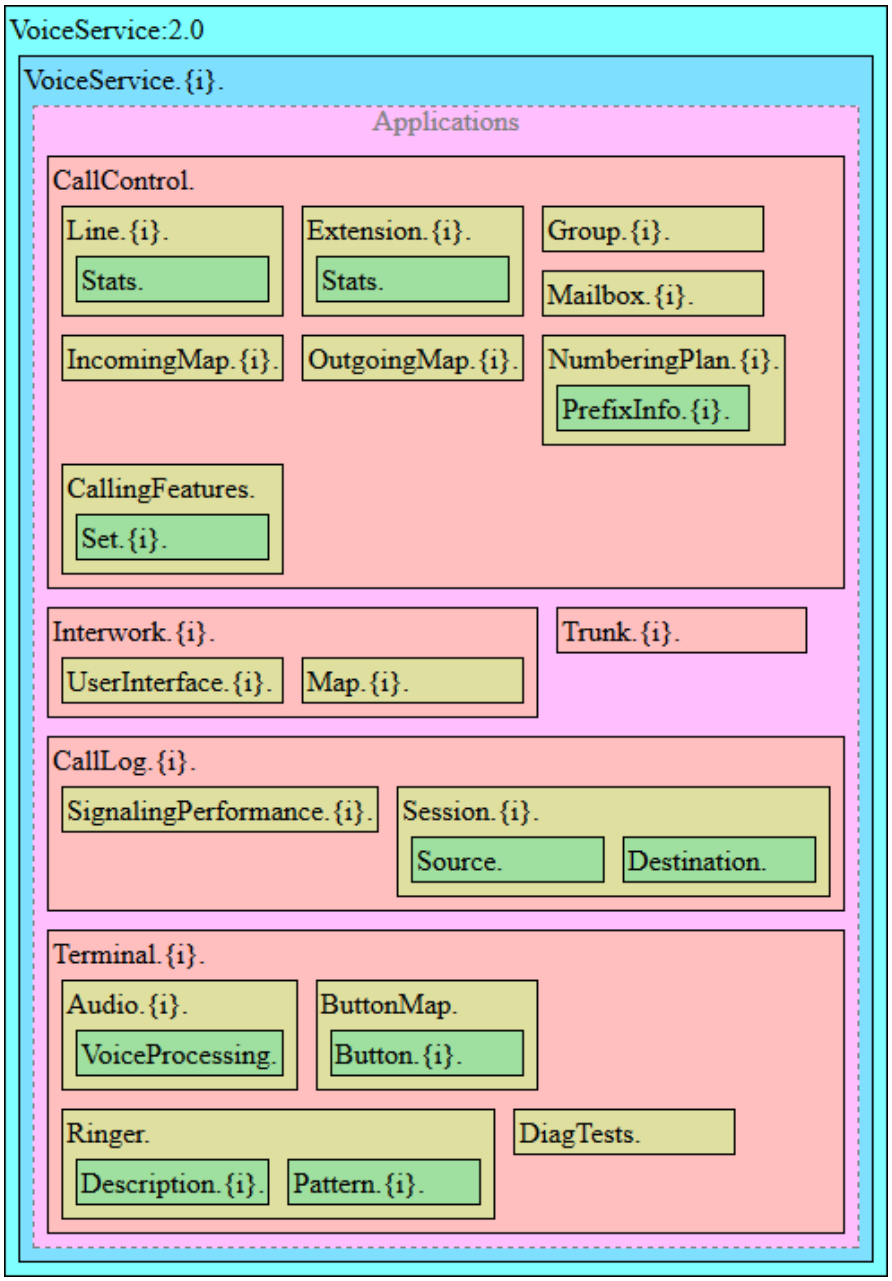

Figure 5 – VoiceService:2 Data Model Structure – Applications

2 References and Terminology

2.1 Conventions

In this Technical Report, several words are used to signify the requirements of the specification. These words are always capitalized. More information can be found be in RFC 2119 [1].

MUST	This word, or the term “REQUIRED”, means that the definition is an absolute requirement of the specification.
MUST NOT	This phrase means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the term “RECOMMENDED”, means that there could exist valid reasons in particular circumstances to ignore this item, but the full implications need to be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED" means that there could exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications need to be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the term “OPTIONAL”, means that this item is one of an allowed set of alternatives. An implementation that does not include this option MUST be prepared to inter-operate with another implementation that does include the option.

2.2 References

The following references are of relevance to this Technical Report. At the time of publication, the editions indicated were valid. All references are subject to revision; users of this Technical Report are therefore encouraged to investigate the possibility of applying the most recent edition of the references listed below.

A list of currently valid Broadband Forum Technical Reports is published at www.broadband-forum.org.

Document		Title	Source	Year
[1]	RFC 2119	*Key words for use in RFCs to Indicate Requirement Levels*	IETF	1997
[2]	TR-069 Amendment 5	*CPE WAN Management Protocol*	BBF	2013
[3]	TR-106	*Data Model Template for TR-069-Enabled Devices*	BBF	2013

	Amendment 7			
[4]	RFC 3261	*SIP: Session Initiation Protocol*	IETF	2002
[5]	RFC 3435	*Media Gateway Control Protocol (MGCP) Version 1.0*	IETF	2003
[6]	RFC 3550	*RTP: A Transport Protocol for Real-Time Applications*	IETF	2003
[7]	TR-104	*Provisioning Parameters for VoIP CPE*	BBF	2005
[8]	RFC 2663	*IP Network Address Translator (NAT)*	IETF	1999
[9]	RFC 6140	*Registration for Multiple Phone Numbers in the Session Initiation Protocol (SIP)*	IETF	2011
[10]	RFC 3263	*Session Initiation Protocol (SIP): Locating SIP Servers*	IETF	2002
[11]	RFC 3611	*RTP Control Protocol Extended Reports (RTCP XR)*	IETF	2003
[12]	ITU-T Y.1540	*Internet protocol data communication service – IP packet transfer and availability performance parameters*	ITU	2011

2.3 Definitions

The following terminology is used throughout this Technical Report.

ACS	Auto-Configuration Server. This is a component in the broadband network responsible for auto-configuration of the CPE for advanced services.
CPE	Customer Premises Equipment.
Directory Number	A distinct number by which a Line is addressed.
Endpoint	A VoIP device that acts as the initiation/termination point for VoIP calls. Examples of Endpoints include VoIP phones and analog terminal adapters (ATAs).
Line	A separately addressable voice line with a distinct Directory Number.
Parameter	A name-value pair representing a manageable CPE parameter made accessible to an ACS for reading and/or writing.
VoIP Profile	Configuration profile that can be shared by many VoIP components.
Session	A single active two-way voice media session. A single Line may support more than one active Session, for example for CPE-provided three-way calling.
Trunk	A Trunk is a circuit between switching systems, such as a PBX, that can carry a specific number of Lines.

Extension In a PBX context, an Extension is a private number that can place or receive calls. An Extension can be a phone connected through a physical port or via VoIP, but also a virtual component such as a group of Extensions.

2.4 Abbreviations

This Technical Report uses the following abbreviations:

AA	Abbreviated Address
AOR	Address-of-Record
ATA	Analog Terminal Adapter
ALG	Application Level Gateway
B2BUA	Back to Bach User Agent
BRI	Basic Rate Interface
CA	Call Anonymous
CAT-iq	Cordless Advanced Technology - internet and quality
CCBS	Call Completion to Busy Subscriber
CCNR	Call Completion on No Reply
CFB	Call Forwarding on Busy
CFT	Call Forwarding Timed
CFU	Call Forwarding Unconditional
CLIR	Calling Line Identification Restriction
CW	Call Waiting
DDI	Direct Dial In
DECT	Digital Enhanced Cordless Telephone
DND	Do Not Disturb
DNS	Domain Name Service
DSP	Digital Signal Processing / Processor
DTMF	Dual-Tone Multi-Frequency
ESBC	Enterprise Session Border Controller
EU	European Union
FQDN	Fully Qualified Domain Name
FXS	Foreign eXchange Subscriber
FXO	Foreign eXchange Office
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISDN	Integrated Services Digital Network
LAN	Local Area Network

MGCP	Media Gateway Control Protocol
MOS	Mean Opinion Score
MSN	Multiple Subscriber Numbering
NT	Network Termination
OCB	Outgoing Call Barring
PBX	Private automatic Branch eXchange
POTS	Plain Old Telephone Service
PRI	Primary Rate Interface
PSO	PSTN Switch Over
PW	Password
RFC	Request For Comment
RTCP	Real-time Transport Control Protocol
RTP	Realtime Transport Protocol
SBC	Session Border Controller
SCF	Service Control Function
SCF	Selective Call Forwarding
SCREJ	Selective Call Rejection
SIP	Session Initiation Protocol
SR	Selective Ringing
SRUP	Sequenced Routing Update Protocol
TE	Terminal Equipment
TISPAN	Telecommunications and Internet Services and Protocols for Advanced Networks
STUN	Simple Traversal of UDP through NATs
TR	Technical Report
TU	TEL URI
URI	Uniform Resource Identifier
VoIP	Voice over IP
WAN	Wide Area Network

3 Technical Report Impact

3.1 Energy Efficiency

TR-104 has no impact on Energy Efficiency.

3.2 IPv6

TR-104 has no impact on IPv6.

3.3 Security

TR-104 has no impact on Security.

3.4 Privacy

Any issues regarding privacy are not affected by TR-104.

4 Architecture

The VoiceService:2 data model defined in this Technical Report consists of a set of data objects covering CPE voice capabilities, voice ports (physical and logical), voice lines, ringers and tones, calling features, call logs and interconnection. It also defines a baseline profile that specifies a minimum level of data model support.

A CPE making use of a VoiceService object MUST adhere to all of the data-hierarchy requirements defined in [3]. In the context of [3], the VoiceService object defined in this specification is a Service Object. As defined in [3], individual CPE MAY contain one or more instances of the VoiceService object. The presence of more than one VoiceService object might be appropriate, for example, where a CPE serves as a management proxy for other non-TR-069 capable VoIP CPE. For example, an Internet Gateway Device might serve as a management proxy for one or more non-TR-069 capable VoIP phones.

4.1 Physical Interfaces

The physical voice ports (ISDN, DECT and POTS) model physical interfaces dedicated to voice on the CPE. Each type of voice port is modeled by a VoiceService:2 data model table, with a row per physical interface instance (e.g. *ISDN.BRI.{i}* for ISDN Basic Rate Interface).

Each physical interface object contains a core set of parameters and objects, which serves as the template for defining physical interface objects within the data model. Physical interface objects can also contain other parameters and sub-objects specific to the type of interface.

The core set of parameters consists of:

- Enable — The administrative state of the physical interface (i.e. boolean indicating enabled or disabled).
- Status — The operational state of the physical interface (i.e. Up, Error, Testing, Disabled).
- Alias — An alternate name used to identify the physical interface, which is assigned an initial value by the CPE but can later be chosen by the ACS.
- Name — The textual name of the interface as assigned by the CPE.
- ToneEventProfile — The tone events configuration profile to use with this physical interface.

4.2 VoIP Network Architecture

Each VoIP connection to the network is modeled by three VoiceService:2 data model tables: *Client*, *Network* and *VoIPProfile*; *Client* and *Network* are protocol specific while *VoIPProfile* is common to all protocols. A row in the *Client* table represents a VoIP client which is responsible for establishing a connection to a VoIP platform and registering the CPE. A row in the *Network* table contains protocol specific network configuration (e.g. platform address, timers…). And finally, a row in the

VoIPProfile table contains configuration that is common to all VoIP protocols. This architecture allows multiple VoIP clients to share the same network configuration and also clients with different protocols to use the same VoIPProfile.

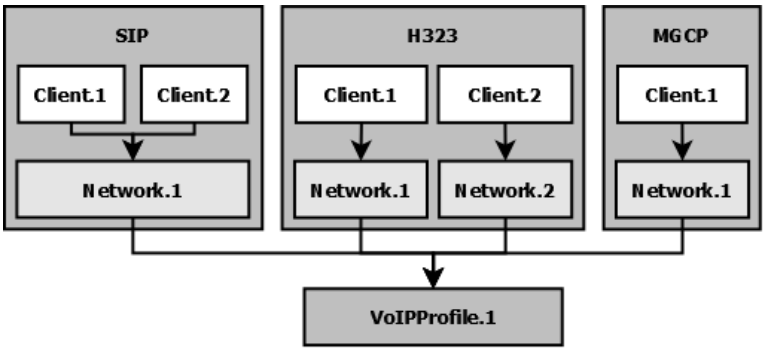

Figure 6 – VoIP Network Configuration Example

Each *Client* object contains a core set of parameters and objects, which serves as the template for defining *Client* objects within the data model. *Client* objects can also contain other parameters and sub-objects specific to the type of client.

The core set of parameters consists of:

- Enable — The administrative state of the client (i.e. boolean indicating enabled or disabled).
- QuiescentMode — Used to maintain in-progress sessions when the client is disabled.
- Status — The operational state of the client (See Figure 7).
- Alias — An alternate name used to identify the client, which is assigned an initial value by the CPE but can later be chosen by the ACS.
- MaxSessions — The maximum number of simultaneous sessions that can be handled by the client.
- Network — A reference to a row in the *Network* table.

The *Status* parameter has the following state diagram:

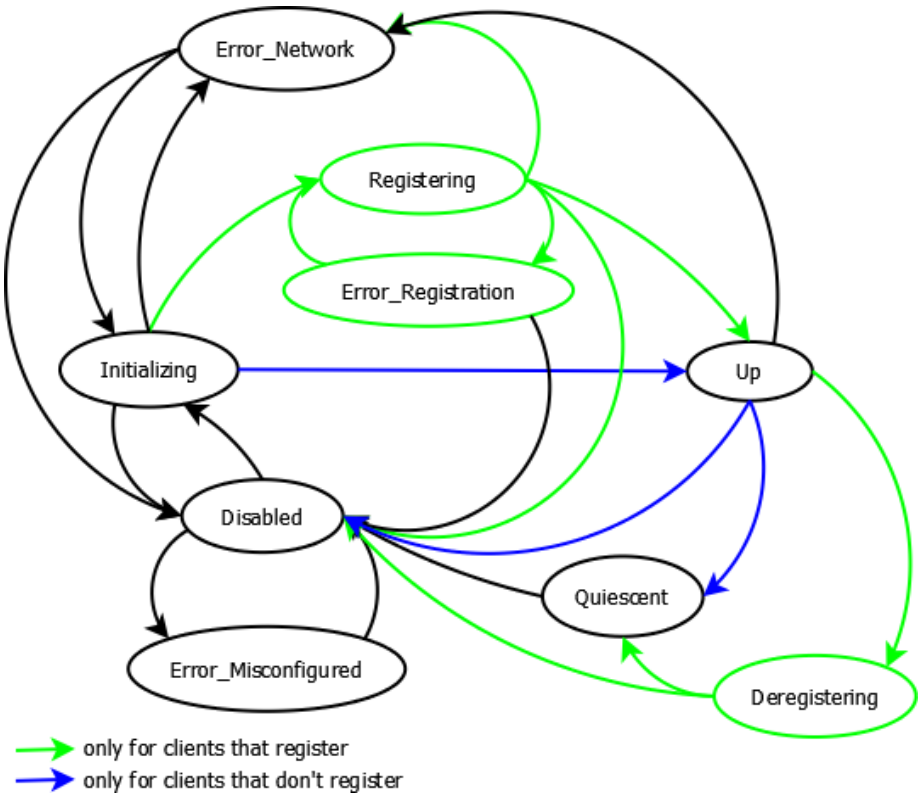

Figure 7 – Client Status Parameter – State Diagram

- Disabled — The client is deactivated.
- Initializing — This is a transitional state that is used to wait for non-VoIP network services (i.e. IP interfaces, DNS, NTP…) to be operational.
- Registering — The client is in registration process.
- Up — The client is fully operational.
- Deregistering — The client is in deregistration process.
- Quiescent — The client has been disabled (i.e. *Enable* has been set to *false*), and is waiting for active VoIP sessions to terminate before transitioning to the disabled state.
- Error_Misconfigured — This state is used when *Enable* is set to *true*, but the client is not operational due a configuration error. In this case, the client MUST be disabled before the configuration can be changed.
- Error_Network — This state is used when a network error occurs during 'Initializing', 'Registering' or 'Up' states. In this state, the client can use a retry mechanism to return in 'Initializing' state or can be disabled.

- Error_Registration — This state is used when an error occurs during registration process. In this state, the client can use a retry mechanism to return in 'Registering' state or can be disabled

Each *Network* object contains a core set of parameters and objects, which serves as the template for defining *Network* objects within the data model. *Network* objects can also contain other parameters and sub-objects specific to the type of network.

The core set of parameters consists of:

- Enable — The administrative state of the network (i.e. boolean indicating enabled or disabled).
- QuiescentMode — Used to maintain in-progress sessions when the network is disabled.
- Status — The operational state of the network (i.e. Up, Resolving, Error_DNS, Error_Other, Disabled).
- Alias — An alternate name used to identify the network, which is assigned an initial value by the CPE but can later be chosen by the ACS
- DSCPMark — Diffserv code point to be used for outgoing signaling packets.
- VLANIDMark — VLAN ID to be used for outgoing signaling packets.
- EthernetPriorityMark — Ethernet priority code to be used for outgoing signaling packets.
- STUNEnable — Enable or disable use of STUN to allow operation through NAT.
- STUNServer — Domain name or IP address of the STUN server.
- NonVoiceBandwidthReservedUpstream — For bandwidth-based admission control, a session can proceed only if there is *NonVoiceBandwidthReservedUpstream bits per second* of upstream bandwidth left available for non-voice traffic.
- NonVoiceBandwidthReservedDownstream — For bandwidth-based admission control, a session can proceed only if there is *NonVoiceBandwidthReservedDownstream bits per second* of downstream bandwidth left available for non-voice traffic.
- MaxSessions — The maximum number of simultaneous sessions that can be handled by the network.
- VoIPProfile — A reference to a row in *VoIPProfile* table.

4.3 User / Network Interconnection

Figure 8 shows the different physical and logical CPE interfaces that are modeled by VoiceService:2.

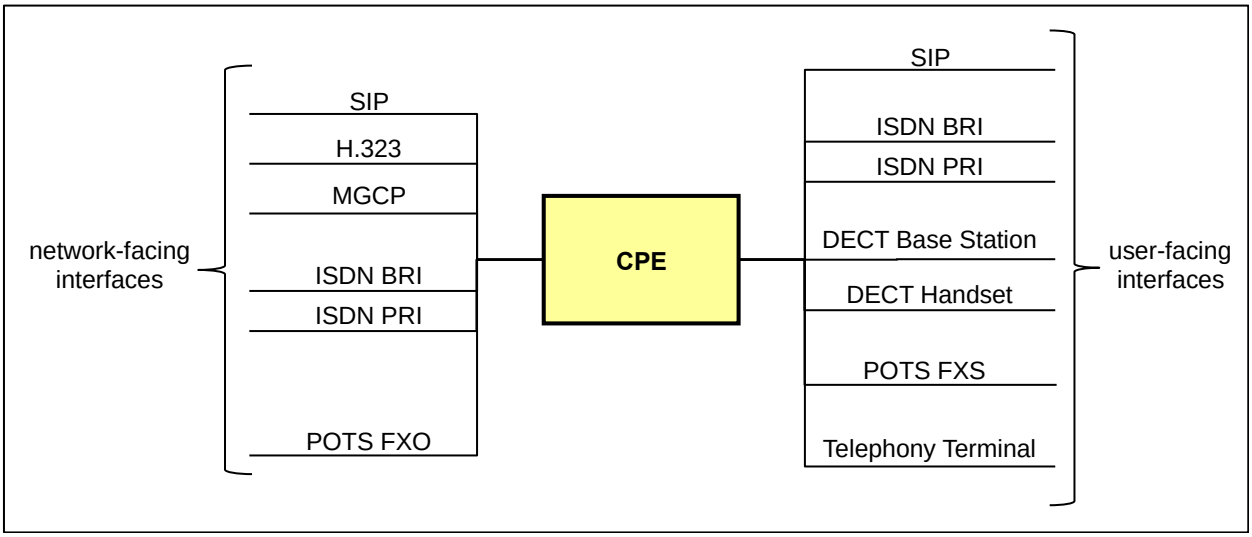

Figure 8 – Generalized TR-104 CPE

A specific CPE use case is realized by mapping one of the network-facing interfaces to one of the user-facing interfaces. Table 1 shows the interface mappings for some common CPE use cases.

Table 1 – CPE Network-to-User Interface Mapping Examples

CPE Use Case	Interface	
	Network-Facing	**User-Facing**
SIP Phone	SIP	Terminal
SIP Terminal Adapter	SIP	POTS FXS
SIP-PBX serving ISDN phones	SIP	ISDN BRI
H.323/PRI Gateway	H.323	PRI ISDN

The VoiceService:2 data model provides two different mapping objects: *CallControl* and *Interwork*.

CallControl is used for more complex CPE use cases such as SIP-PBX or SIP-based key systems; use cases that require some level of feature control and complex incoming/outgoing call routing algorithms. Specifically, *CallControl* maps network-facing *Line* objects to user-facing *Extension* objects. Lines and Extensions model the calling features at the "application-layer" of the telephony OSI stack and in turn link to objects that model a specific physical or logical interface.

Interwork is used for simpler CPE use cases such as Gateway devices and simple terminal devices; use cases that don't require feature control. *Interwork* maps the network-to-user objects that model

the actual physical and logical interfaces of the CPE. For example, *Interwork* maps a network-facing *SIP.Client* to a user-facing *SIP.Registrar.Account* to create a SIP-ALG.

The following CPE use case examples are modeled using *CallControl*:

- DECT CAT-iq 2.0
- ISDN with MSN
- Local PBX functionality
- Voicemail
- Least Cost Routing
- Local implementation of calling features

The following CPE use case examples are modeled using *Interwork*:

- VoIP Gateway
- Basic SIP Terminal

4.3.1 Call Control

4.3.1.1 Lines and Extensions

Typically, the role of the *CallControl* application is to handle incoming and outgoing calls and to route them to their destination. This is realized by linking *CallControl.Line* objects to *CallControl.Extension* objects with *CallControl.IncomingMap* and *CallControl.OutgoingMap* tables.

The *CallControl.Line* table represents all voice lines used by the CPE to send or receive calls to the Network. Depending on the underlying technology, a *Line* object can be a physical line (e.g. ISDN or FXO), a logical line (e.g. SIP or H.323) or can be part of a *Trunk*. The *Provider* reference determines the type of the *Line*.

Table 2 – List of Line types

Provider parameter	Type	Description
ISDN.BRI or *ISDN.PRI*	Physical	*Line* using an ISDN physical interface.
POTS.FXO	Physical	*Line* using an FXO physical interface.
SIP.Client	Logical	*Line* based on a SIP client.
H323.Client	Logical	*Line* based on a H.323 client.
Trunk	Physical or Logical	One of the *Line* instances contained in a *Trunk*.

The *Extension* table represents all local numbers directly managed by the CPE. An extension can place or receive calls to the voice network using a line or to another extension depending on the *CallControl* configuration. Like the *Line* object, an *Extension* can be a physical extension (e.g. ISDN, FXS or DECT) or a logical extension (e.g. SIP or Group). The *Provider* parameter determines the type of the *Extension*.

Table 3 – List of Extension types

Provider parameter	Type	Description
ISDN.BRI or *ISDN.PRI*	Physical	Extension using an ISDN physical interface.
POTS.FXS	Physical	Extension using an FXS physical interface.
DECT.Portable	Physical	Extension using a portable attached to DECT base.
Terminal	Physical	Special extension which represents the CPE itself.
SIP.Registrar.{i}.Account	Logical	Extension based on a locally-registered SIP client.
CallControl.Group	Logical	Virtual extension which is a group of other extensions.

4.3.1.2 Calling Features

CallingFeatures is a term commonly used in the industry to name extra call-related functionalities that are offered to the subscribers.

In pre-VoIP times the operator implemented these functionalities in the central office switch. Also PBXs typically offer these calling features.

Since the introduction of VoIP based services there is an ongoing battle on where these features are implemented. In scenario that are more PBX-like where a device is providing voice service for multiple users in the home or office and where each user has his preferences.

In interwork scenario (non-PBX), calling features are usually not required. Protocol specific aspects can be configured using protocol event filters.

The *CallingFeatures* object is meant for storing the state of these services in the device. The user can typically modify these features on the device using DTMF based feature codes or via specific protocol messages (ISDN, DECT).

4.3.1.3 Numbering Plan

The *CallControl.NumberingPlan* table is used to configure how the device reacts on particular numbers. Calls initiated from an extension will use the associated *NumberingPlan* instance for call routing decisions.

The numbering plan is only used for the exceptions to default routing. Default routing, which does not need to be configured, includes:

- Routing to extensions based on extension numbers.
- Last resort routing to the *CallControl.Line* object(s) found in the *CallControl.OutgoingMap* table for the *CallControl.Extension*.

The following use cases are supported by the numbering plan:

- DTMF feature codes to configure calling features.
- Special routing (ex. 112 or 911 via line connected to FXO).
- Routing Codes allowing the user to select an outgoing line other than default.
- Least Cost Routing: select different lines based on prefix.
- Mailbox access (facility action added).
- Short numbers / phonebook.

When DTMF based feature codes (ex. "*21*456789123#") are used, the numbering plan is used to configure which codes are used for which services. The facility action parameter is used to define the action to perform. The codes used are often country or operator specific.

For call routing, the parameter *CallControl.NumberingPlan.{i}.PrefixInfo.{i}. FacilityActionArgument* is used to select the line used for outgoing calls.

The table below contains examples of some common *NumberingPlan* use cases.

Index	PrefixRange	PrefixMinNumberOfDigits	PrefixMaxNumberOfDigits	NumberOfDigitsToRemove	PosOfDigitsToRemove	DialTone	FacilityAction	FacilityActionArgument
1	112	3	3	0	0		LINE_INVOKE	Line.1
2	*21*	4	4	4	0		CFU_REGISTER	
3	#21#	4	4	0	0		CFU_DEACTIVATE	
4	##21#	5	5	0	0		CFU_ERASE	
5	*21#	4	4	0	0		CFU_ACTIVATE	
6	*#21#	5	5	0	0		CFU_INTERROGATE	
7	*111#	5	20	5	0		LINE_INVOKE	Line.1
8	*112#	5	20	5	0		LINE_INVOKE	Line.2
9	*113#	5	20	5	0		LINE_INVOKE	Line.3
10	**600	5	5	5	0		MAILBOX_INVOKE	
11	00	2	20	0	0	Tone.4	CP_INVOKE	1122

Index 1 defines a specific line (ex. FXO) to be used for emergency calls.
Index 2-6 defines the commonly used (in the EU) ETSI Standard codes for “call forward unconditional”.
Index 7-9 defines a set of proprietary codes for selecting the outgoing line.
Index 10 defines a proprietary code to listen to messages on the mailbox.
Index 11 indicates that a carrier select prefix 1122 should be used for international calls (European standard). A special dial-tone is presented to the user after dialing 00 to inform him of this carrier change.

4.3.1.4 CallControl Use Cases

4.3.1.4.1 Advanced SIP Terminal

An advanced terminal is a SIP Endpoint that uses more than one *Line*. It can also have advanced features such as an internal mailbox. In this case, *CallControl* has to be used with one *CallControl.Extension* object representing the terminal.

Special buttons in *Terminal.ButtonMap* table can be used to:

- Invoke a specific line to use
- Invoke the mailbox
- Activate/deactivate do-not-disturb
- …

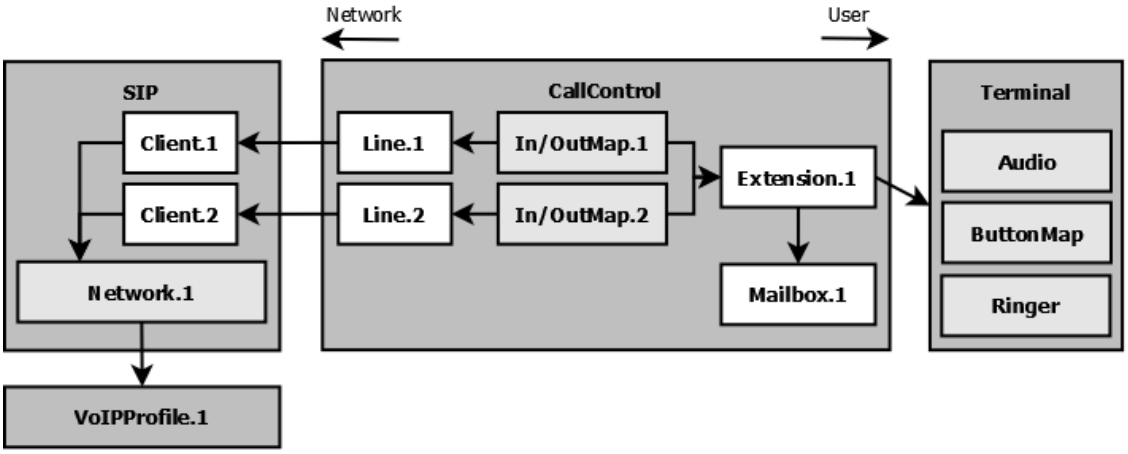

Figure 9 – Advanced SIP Terminal Use Case

4.3.1.4.2 ISDN MSN

Using the MSN feature on ISDN interfaces can be modeled using the *Line* and *Extension* objects from *CallControl*. For each MSN number on a TE interface a *CallControl.Line* instance is assigned to the ISDN interface. For each MSN number on a NT interface *CallControl.Extension* instance is assigned to the ISDN Interface.

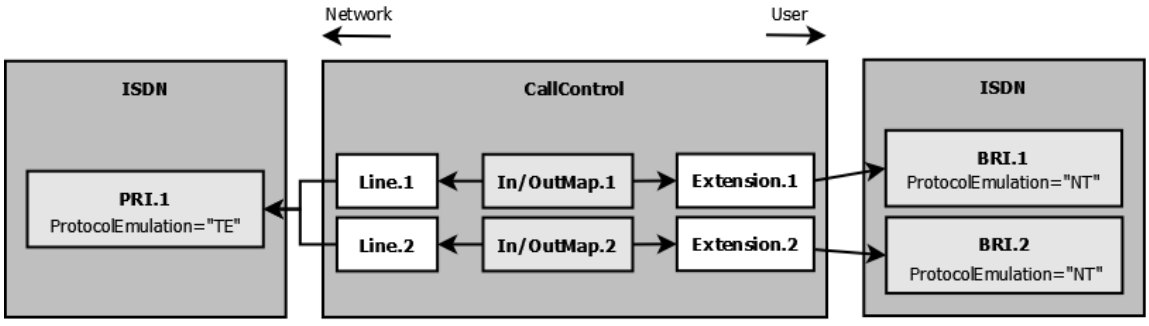

Figure 10 – ISDN MSN Use Case

4.3.1.4.3 SIP Trunk with PBX

The device has two FXS interfaces and a build in DECT CAT-iq base station and it supports a registrar for SIP phones.

The default configuration of the device has *CallControl.Extension.1* with number "**01" linked to *POTS.FXS.1* and *CallControl.Extension.2* with number "**02" to *POTS.FXS.2*. *CallControl.Extension.3* with number "**09" is linked to a group that points to all other extensions.

The default configuration also contains a *CallControl.NumberingPlan* that has a default mapping for all the supported facility actions and for a mailbox application. All *CallControl.Extension* objects reference this numbering plan.

The device has built-in logic that auto-creates a *CallControl.Extension* object each time the user peers a DECT portable with the base. The number range "**11" to "**19" is reserved for these extensions. Whenever the GUI is used to peer a SIP phone with the device, an extension with number range "**21" to "**29" is assigned to it.

All *CallControl.Extension* objects automatically reference an auto-created *CallingFeatures.Set* object and an auto-created *CallControl.Mailbox* object.

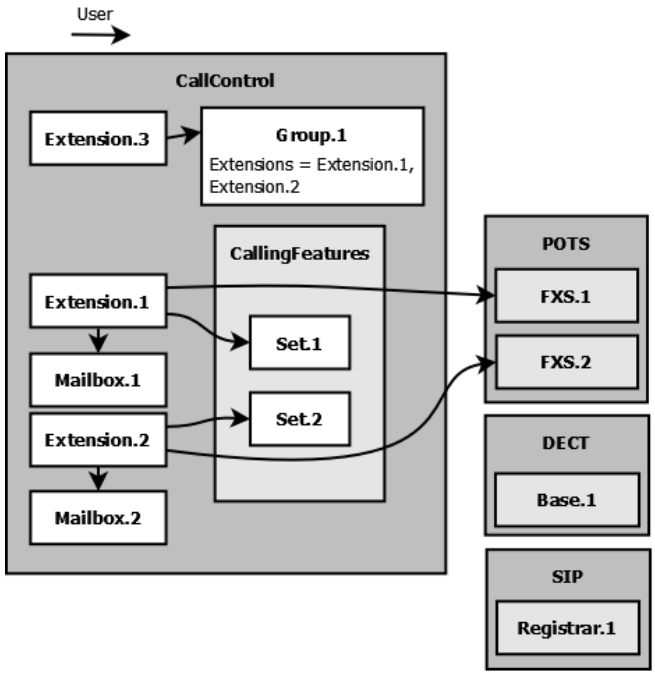

Figure 11 – SIP Trunk use case: Initial State

When the device is installed for the first time the ACS configures the following items:

1. Create a *VoIPProfile* object and configure the VoIP media specific parameters.
2. Create a *SIP.Network* object and configure the SIP servers of the service provider.
3. Create a *SIP.Client* object with credentials to authenticate on the platform.

4. Create a *Trunk* object to define the number range. The device auto-creates a *CallControl.Line* object for each of the numbers in the number range.
5. Create *CallControl.IncomingMap* entries linking each line to *CallControl.Extension.3*.
6. Create *CallControl.OutgoingMap* entries linking each extension to *CallControl.Line.1*.

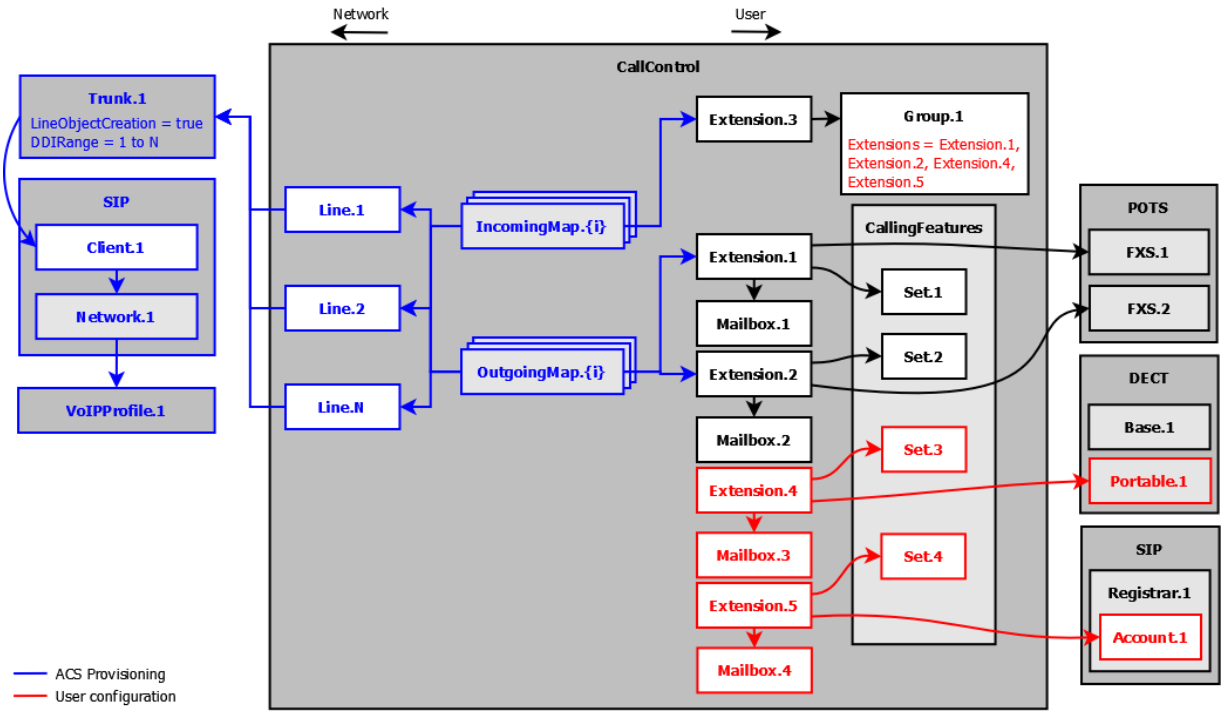

Figure 12 – SIP Trunk use case: After ACS Provisioning and User Configuration

The service provider also provides a self-care portal that allows the user to change the mapping between lines and extensions. The ACS stores this information in the subscriber profile and configures this on the device.

In case of device replacement the ACS can reconfigure the settings including the changes made by the user on the self-care portal.

4.3.2 Interwork

The *Interwork* table provides a CPE with the ability to map various Network connections (e.g. *Trunk*, *SIP.Client* objects) to various User connections (e.g. *ISDN.PRI*, *Terminal*, *SIP.Registrar.{i}.Account* objects). Unlike the *CallControl* table, the *Interwork* table does not require separate inbound and outbound directional mapping . Also, *Interwork* is used to model devices that don't support call features, and therefore it does not map feature-related objects such as *Line* and *Extension*.

The *Interwork* table contains two mode parameters that indicate how certain Network connection objects and the User connection objects are created; whether dynamically by the CPE or statically by the ACS.

- **NetworkConnectionMode:** indicates whether the network-facing *Client* object and its parameters are statically provisioned by the ACS or dynamically created by the CPE. It also indicates whether or not the CPE supports a registration procedure to dynamically convey its network-facing signaling address to the service provider network.
- **UserConnectionMode:** indicates whether the user-facing objects and parameters that contain the signaling address of the external user device are statically provisioned by the ACS or dynamically learned by the CPE. It also indicates whether or not an externally connected user device supports a registration procedure to convey its signaling address to the CPE.

The *Interwork* table is designed to provide a configurable and flexible interworking capability for current and future use cases.

If non-empty, the *Interwork.{i}.ProxyServer* defines a proxy server that connects the interwork functionality to the Network.

This section will detail the main supported use cases.

4.3.2.1 SIP Terminal

When the CPE is an Endpoint, the *Terminal* object has to be used. The following example illustrates a simple SIP Endpoint:

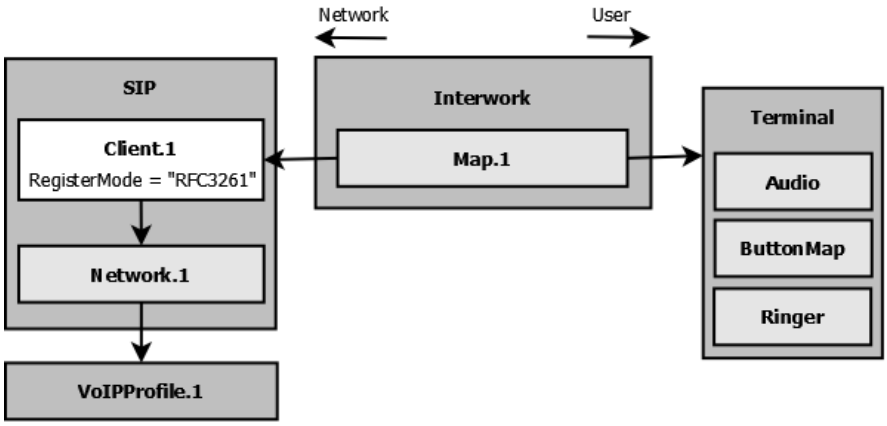

Figure 13 – Simple SIP Terminal Use Case

4.3.2.2 SIP Gateway with FXS and DECT Ports

The device has two FXS interfaces and a built-in DECT CAT-iq base station. On the network side, a SIP client is configured.

The *Interwork.{i}.Map* table is used to associate the SIP client with all the physical interfaces.

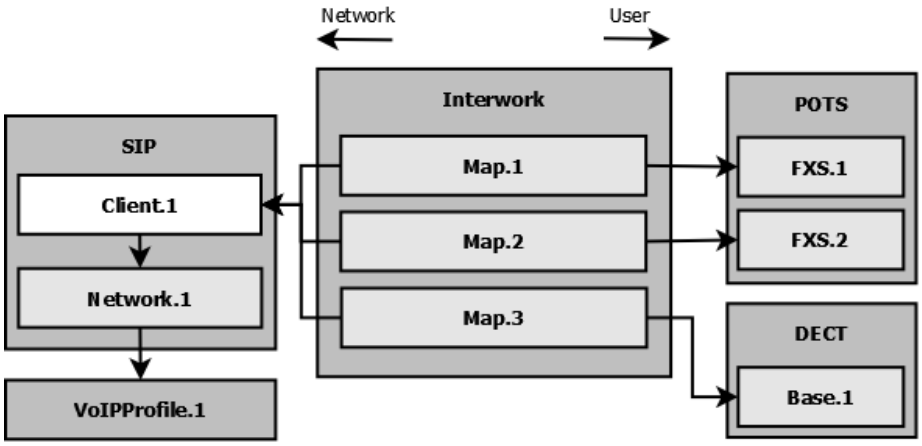

Figure 14 – Simple SIP Gateway Use Case

4.3.2.3 SIP-ISDN PBX Plug Use Case

In this case, the CPE has a SIP client which is acting as a Trunk (*RegisterMode* = "TISPAN"). As all PBX functionality is handled by an external PBX, the *Interwork.{i}.Map* table connects the SIP Trunk to this PBX and the parameter *LineObjectCreation* of the Trunk prevents the creation of *CallControl.Line* objects which are not managed by the CPE.

In the following example, the external PBX is using SIP to register to the CPE with the first account of the SIP Registrar:

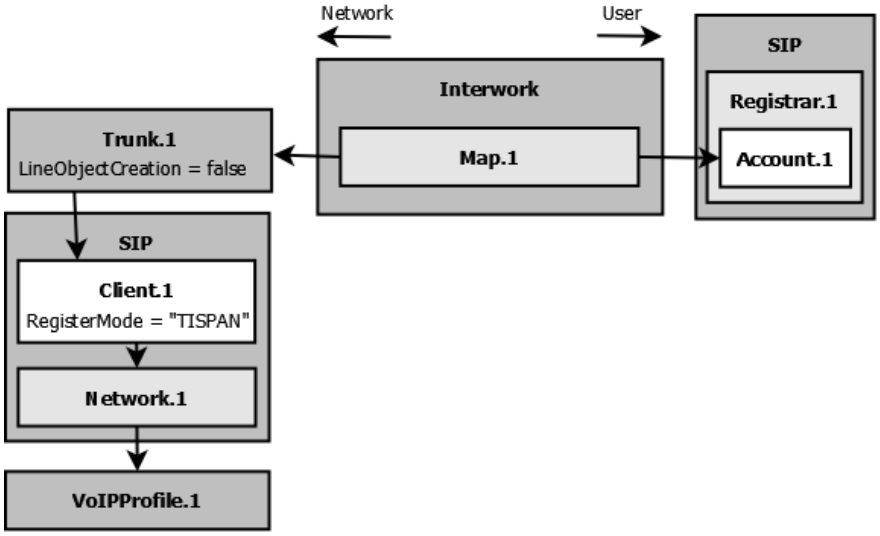

Figure 15 – SIP PBX Plug Use Case

Same principle, but this time this is an ISDN PBX plugged on the first PRI interface:

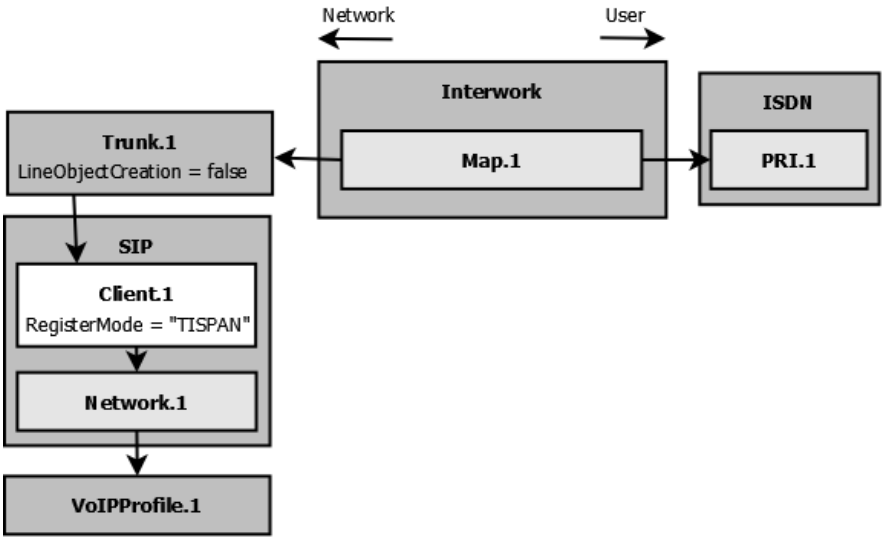

Figure 16 – ISDN PBX Plug Use Case

4.3.2.4 ESBC – Enterprise Session Border Controller

The Enterprise Session Border Controller (ESBC) is a SIP Gateway device that is designed to simplify the initial deployment and ongoing management of SIP Trunk and Hosted IP-Centrex services to business customers. As shown in Figure 17, the ESBC sits at the boundary between the service provider and enterprise network, providing a well-defined demarcation point between the two networks.

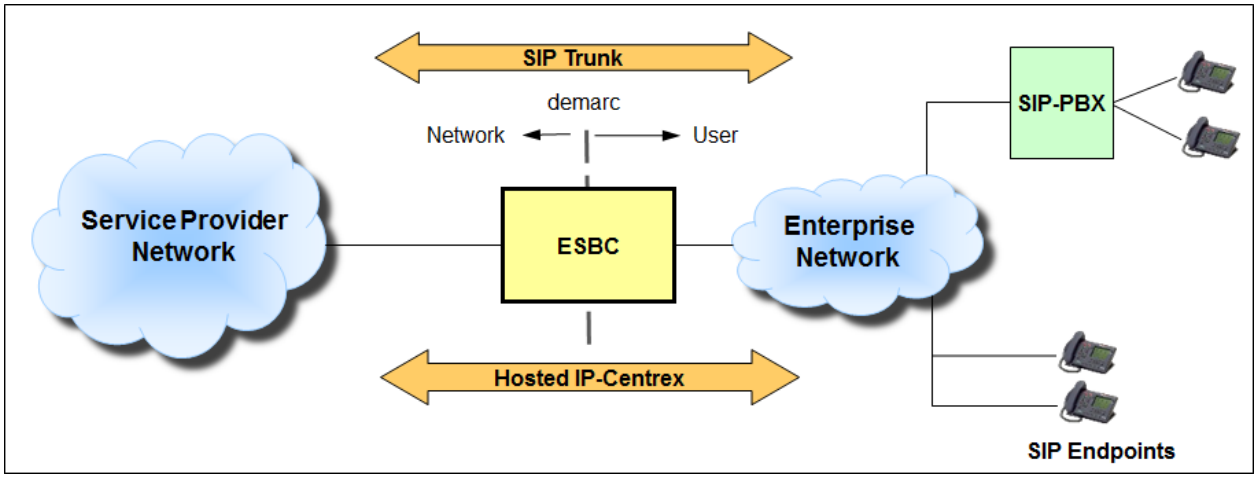

Figure 17 – ESBC Network Architecture Overview

The ESBC supports multiple functions, including:

- **SIP Interworking**
  Normalizes the wide variety of SIP signaling protocols supported by currently deployed enterprise SIP Endpoints (e.g., SIP phones, SIP-PBXs) into a well-defined interface that is compatible with the service provider network.
- **SIP-aware NAT/Firewall**
  Provides IP address:port interworking between the service provider's public IP address space and the enterprise's private IP address space. It also applies firewall rules, to allow only authorized real-time communication traffic to traverse the service provider / enterprise network boundary.
- **Fault isolation and recovery**
  Provides enhanced fault detection and reporting capabilities that speed up the detection, isolation, and resolution of service-affecting failures.

4.3.2.4.1 Internal Structure of ESBC

Figure 18 shows a simplified view of the internal structure of an ESBC. As shown in the diagram, the ESBC contains multiple SBCs, where a single SBC serves a group of users that are related in some way. For example, in a multi-business deployment, each SBC could be assigned to a different business.

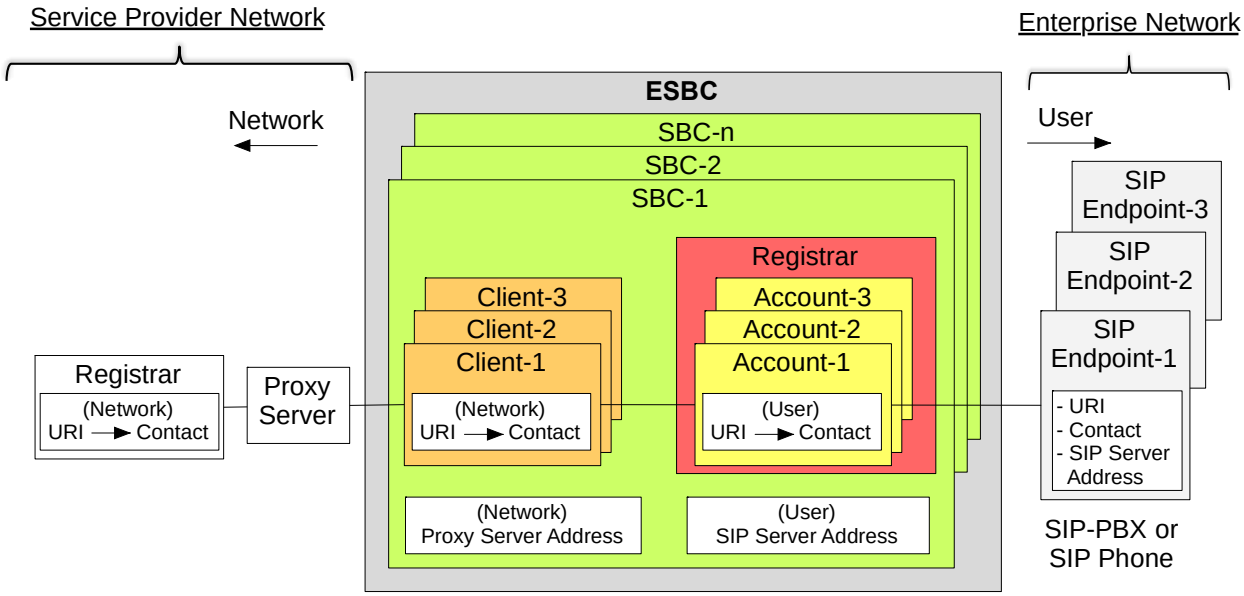

Figure 18 – Internal Structure of ESBC

In the example shown in Figure 18, a single enterprise network containing three SIP Endpoints is served by SBC-1 (where a SIP Endpoint is a SIP-PBX or a SIP phone). The SBC supports a *SIP.Registrar* function that contains multiple *SIP.Registrar.Account* objects; one account per enterprise SIP Endpoint. Each account maintains the User name-to-location (i.e. URI-to-Contact) binding of the associated enterprise SIP Endpoint. The SBC maps each *SIP.Registrar*. *Account* object to a *SIP.Client* object that contains the Network name-to-location binding for the same SIP Endpoint. The *SIP.Registrar* function in the service provider network maintains the same Network name-to-location binding information.

The SIP Endpoints in the enterprise are configured with a next-hop proxy SIP server address that points to the ESBC User interface (i.e. from the SIP Endpoint's perspective, the ESBC is its next-hop proxy). The ESBC SBC is configured with this same User SIP server address where it listens for incoming SIP requests from the enterprise SIP Endpoint. The SBC is also configured with the Network proxy server address of the service provider SIP proxy assigned for this enterprise.

The ESBC acts as a twice-NAT as defined in [8]. For example, for upstream SIP requests sent from the enterprise SIP Endpoint to the service provider network, the ESBC maps both the IP destination and source addresses of the request:

- The destination address from the ESBC's User SIP server address to the Network proxy server Address.
- The source address from User Contact address to Network Contact address

The ESBC performs symmetrical Network-to-User address mappings of the source and destination addresses for downstream requests sent from the service provider network to the enterprise SIP Endpoint.

The ESBC is required to operate in a number of different deployment modes that are primarily driven by two factors:

- **The Type of Service:** whether the service provider is providing SIP Trunking service or Hosted IP-Centrex service to the enterprise customer (in other words, whether the enterprise network contains SIP phones or SIP-PBXs)
- **Name-Location Discovery Mechanism:** whether the name-location binding of the enterprise SIP Endpoints is discovered dynamically via SIP registration, or provisioned statically by the ACS.

For example, for hosted IP-Centrex service where the enterprise network contains SIP phones, the ESBC creates the *SIP.Registrar.Account* / *SIP.Client* object pairs containing the User and Network name-location address bindings dynamically as the SIP phones register. On the other hand, for SIP trunking service operating in the Static mode (where the SIP-PBX doesn't register), the ESBC *SIP.Registrar.Account* / *SIP.Client* object pairs and their respective User and Network name-location bindings are created statically by the ACS.

4.3.2.4.2 Data Model Support of ESBC

In the VoiceService:2 data model, the *Interwork* table models the ESBC SBCs. Each *Interwork* table entry is configured with four unique mode parameters to indicate how the *SIP.Registrar.Account* and *SIP.Client* objects are created (whether dynamically or statically, etc). These four parameters are:

- **NetworkConnectionMode:** indicates how the Network name-to-location binding of the enterprise SIP endpoint is obtained by the service provider network; whether conveyed by the ESBC via some form of SIP registration, or statically provisioned.
- **UserConnectionMode:** indicates how the User name-to-location binding of the enterprise SIP Endpoint is obtained by the ESBC; whether via some form of SIP registration, or statically provisioned.
- **E164Mode:** indicates whether or not the ESBC contains E.164 mapping information to route incoming calls to the correct SIP-PBX.
- **NetworkAuthenticationChallengeMode:** indicates how the ESBC treats Digest authentication challenges received from the service provider network; whether the ESBC passes them transparently through to the enterprise SIP endpoint, or responds to them on behalf of the enterprise Endpoint.

More details on these mode parameters and how they drive the *Interwork* table entry behavior can be found in Appendix I.

The following subsections show the VoiceService:2 data model support for the two primary use cases: Hosted IP-Centrex service, and SIP trunking service.

4.3.2.4.2.1 Hosted IP-Centrex Service

Figure 19 shows the VoiceService:2 data model representation for the hosted IP-Centrex service case, where the enterprise network contains SIP phones (either dedicated hard-phone devices, or soft phones) that register with the service provider network.

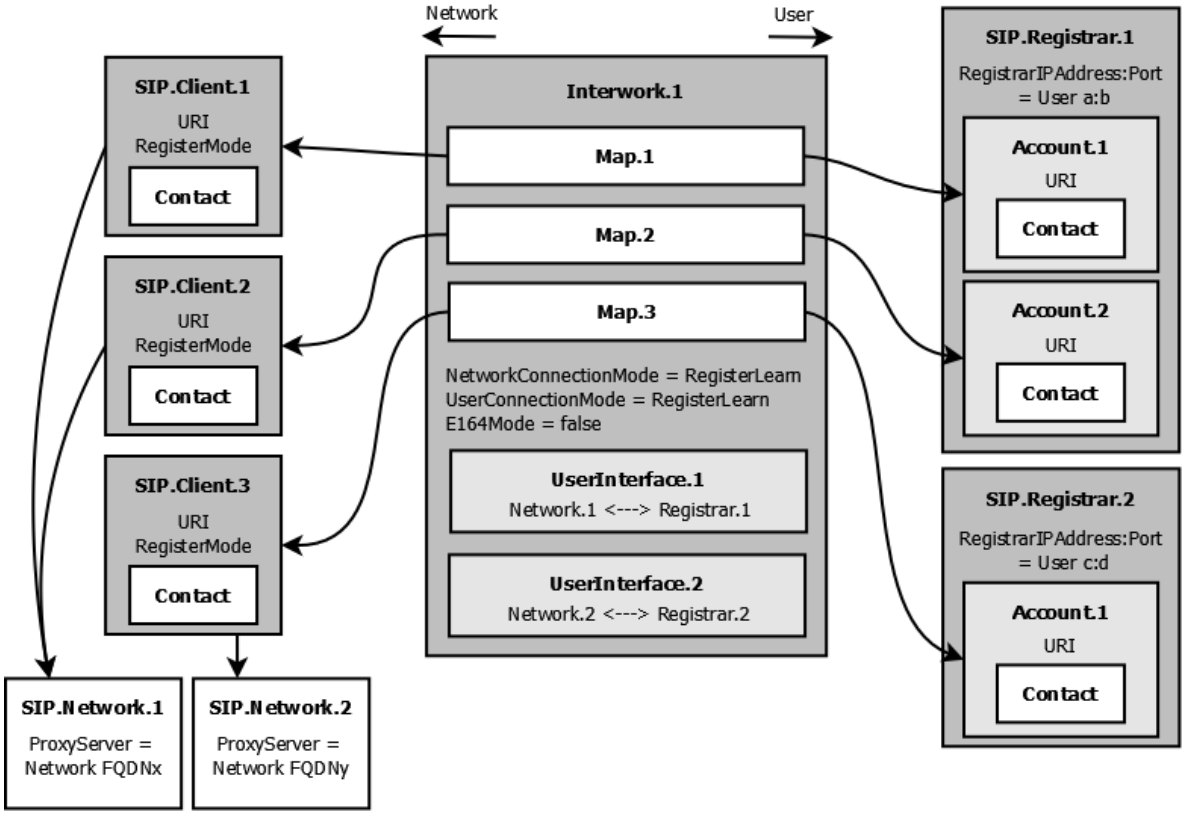

Figure 19 – Hosted IP-Centrex Use Case

Before service turn-up, the ACS pre-provisions the ESBC with a single *Interwork* object, *Interwork.1*. *Interwork.1* is provisioned with the three mode parameters that indicate the target deployment use case and hence, whether the various objects and parameters are statically provisioned by the ACS or dynamically created by the ESBC (Please refer to Appendix I for a description of the fourth mode parameter – *NetworkAuthenticationChallengeMode*). In this case the *NetworkConnectionMode* and *UserConnectionMode* are both set to a value of *RegisterLearn*, which indicates that this is a hosted IP-Centrex case where the *SIP.Registrar.Account* and *SIP.Client* objects (and their contained parameters) are created dynamically when the SIP phones register.

The ACS provisions the *Interwork.1* object with two *UserInterface* objects, where each entry maps a *SIP.Registrar* object to a *SIP.Network* object. Provisioning multiple *Registrar* / *Network* object pairs enables the enterprise signaling traffic load to be distributed across multiple Proxy Server entry points into the service provider network. In this example, the ACS also creates two *SIP.Network* and two *SIP.Registrar* objects. Each *Network* identifies the *ProxyServer* address where the ESBC should send dialog-initiating requests to the service provider network. Each *SIP.Registrar*

contains the *RegistrarIPAddress*:*RegistrarPort* where the registrar listens for incoming dialog-initiating requests from the SIP Endpoints.

In this example, there are three registering SIP phones; two register to the listening IP address:port of *SIP.Registrar.1*, and one to the listening IP address:port of *SIP.Registrar.2*. On completing each registration transaction, the ESBC dynamically creates a *SIP.Registrar.Account* / *SIP.Client* pair for the registering Endpoint, and links the pair via a dynamically created *Interwork.Map* object. The *SIP.Registrar.Account* object contains the URI and User contact information learned from the REGISTER request. The *SIP.Client* object contains the same URI information, and Network contact addresses dynamically assigned by the ESBC for the registering Endpoints. Finally, each *SIP.Client* object is linked to the appropriate *SIP.Network* table entry (based on information in the pre-provisioned in the *Interwork.1.UserInterface.1* instance).

4.3.2.4.2.1.1 ESBC Routing of Hosted SIP Phone Requests

On receiving a dialog-initiating request on its Network interface from the service provider network, the ESBC finds the *SIP.Client.{i}.Contact* object that matches the incoming Request-URI, and then follows the *Interwork.{i}.Map* linkage to the mapped *SIP.Registrar.{i}.Account.{i}.Contact* object. The ESBC then uses the account contact information to update SIP request before forwarding to the target SIP phone.

On receiving a dialog-initiating request on a listening port of *SIP.Registrar* from the SIP phone, the ESBC finds the *SIP.Registrar.{i}.Account.{i}.Contact* object within *SIP.Registrar* that matches the received Contact header field, and then follows the *Interwork.{i}.Map* linkage to the mapped *SIP.Client* object. The ESBC uses the information in the *SIP.Client* object (and its linked *SIP.Network* object) to update the request before forwarding to the service provider network.

4.3.2.4.2.2 SIP Trunking Service with Single-Registering SIP-PBX

Figure 20 shows the VoiceService:2 data model representation for SIP Trunking service case, where the enterprise network contains three SIP-PBXs. Each SIP-PBX performs a single SIP registration transaction to register its multiple subordinate SIP phones (aka Single-Registering SIP-PBX).

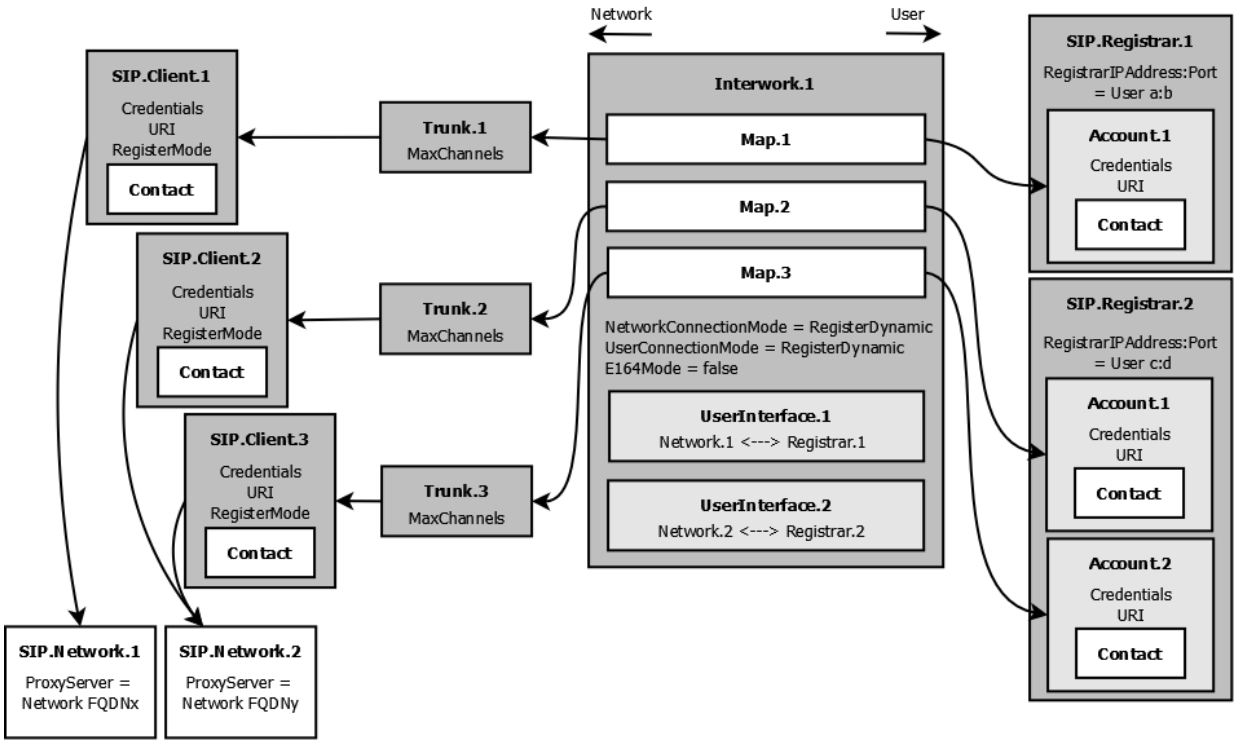

Figure 20 – SIP Trunk with Single-Registering SIP-PBXs Use Case

As in the hosted case, the ACS creates a single *Interwork* object, which is provisioned with the three mode parameters. The *NetworkConnectionMode* and *UserConnectionMode* are set to *RegisterDynamic*. This indicates that the *SIP.Registrar.Account* and *SIP.Client* objects are statically provisioned, but their respective User / Network contact addresses are dynamically discovered / allocated. Also as in the hosted case, the ACS provisions two *Interwork.{i}.UserInterface* objects, along with the associated *SIP.Registrar* and *SIP.Network* objects (the ACS could have provisioned only one *UserInterface* object, but we're showing the case where multiple objects are provisioned to illustrate how call signaling traffic can be distributed across multiple proxy server entry points into the service provider network).

Instead of mapping the registrar directly to the client as was done in the hosted case, the ACS inserts a *Trunk* object in the Registrar-to-Client linkage. The *Trunk* object contains information associated with a SIP *Trunk* instance, such as the simultaneous call capacity the Trunk is allowed to carry.

In this example, there are three SIP-PBXs in the enterprise network; one assigned the listening IP address:port of *SIP.Registrar.1*, and two to the listening IP address:port of *SIP.Registrar.2*.

4.3.2.4.2.2.1 ESBC Routing of Single-Registering SIP-PBX Requests

On receiving a dialog-initiating request on its Network interface from the service provider network, the ESBC finds the *SIP.Client.{i}.Contact* object that matches the incoming Request-URI, and then follows the *Interwork.{i}.Map* linkage to the mapped *SIP.Registrar.{i}.Account.{i}.Contact* object. The ESBC then uses the account and contact information to update SIP request before forwarding to the target SIP-PBX.

On receiving a dialog-initiating request on a listening port of *SIP.Registrar* from the SIP-PBX, the ESBC finds the *SIP.Registrar.{i}.Account* object whose URI and contact matches the received From and Contact header fields, and then follows the *Interwork.{i}.Map* linkage via the mapped *Trunk* to the linked *SIP.Client* object. The ESBC uses the information in the *SIP.Client* object (and its linked *SIP.Network* object) to update the request before forwarding it to the service provider network.

4.3.2.4.2.3 SIP Trunking Service with Multi-Registering SIP-PBX

Figure 21 shows the ESBC VoiceService:2 data model representation for the case where the enterprise contains a multi-registering SIP-PBX; i.e., a SIP-PBX that initiates multiple registration transactions towards the service provider network; one per subordinate SIP phone served by the SIP-PBX. The ESBC in turn generates a single SIP registration transaction towards the service provider network.

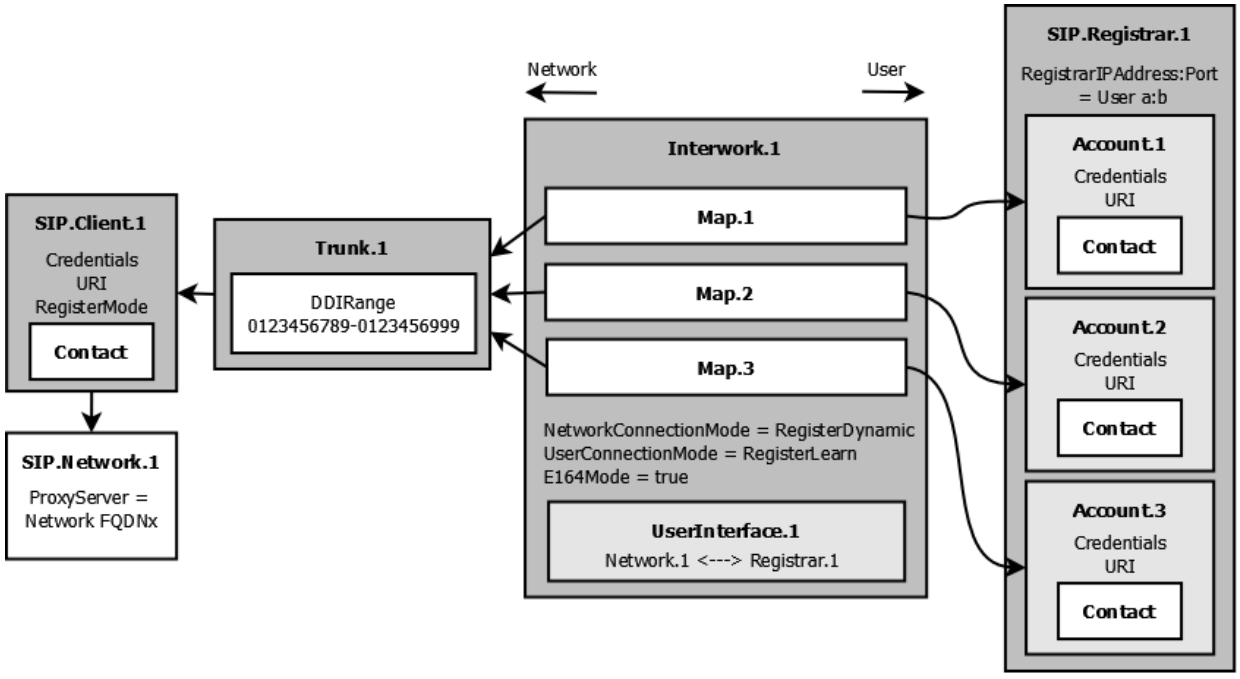

Figure 21 – SIP Trunking Service with Multi-Registering SIP-PBXs Use Case

The ACS creates a single *Interwork* object, which is provisioned with the three mode parameters. The combination of *NetworkConnectionMode = RegisterDynamic* and *UserConnectionMode = RegisterLearn* identifies this specific use case, where the ESBC supports a single registration instance toward the service provider network on behalf of a multi-registering SIP-PBX.

The ACS first pre-provisions the *Interwork*, *Trunk*, *SIP.Client*, *SIP.Network* and *SIP.Registrar* objects, and links them as shown in Figure 21. Then, for each new registration transaction received from the multi-registering SIP-PBX, the ESBC dynamically creates the *SIP.Registrar*. *Account* and *SIP.Registrar.Account.{i}.Contact* objects for the registration transaction and adjusts the *Trunk{i}.DDIRange* to map to the set of accounts associated with the registering SIP-PBX.

4.3.2.4.2.3.1 ESBC Routing of Static SIP-PBX Requests

The routing of dialog-initiating SIP request received on its Network interface from the service provider network is similar to the single-registering SIP-PBX use case described earlier, except that the ESBC uses the *Trunk.DDIRange* parameter to find the linked *SIP.Registrar* object associated with the target SIP-PBX. It then uses the called E.164 number from the incoming message to find the target *SIP.Registrar.Account* object and then updates and routes the request to the SIP-PBX.

5 Parameter Definitions

The normative definition of the VoiceService:2 data model is split between several DM Instance documents (see TR-106 [3] Annex A) and is published at http://www.broadband-forum.org/cwmp. For a given revision of the data model, the corresponding TR-104 Issue 2 XML document defines the VoiceService:2 model itself and imports additional components from the other XML documents listed. Each TR-104 Issue 2 HTML document is a report generated from the XML files, and lists a consolidated view of the VoiceService:2 data model in human-readable form.

Appendix I. Enterprise Session Border Controller

I.1 ESBC Deployment Use Cases

The ESBC supports a number of different deployment scenarios that are determined by the combination of ESBC Network and User name-to-location binding procedures, which are in turn driven by the service and type of SIP Endpoint deployed in the enterprise.

The ESBC use cases are summarized in Table 4.

Table 4 – ESBC Deployment Use Cases

Type of Enterprise SIP Endpoint	Network Name-Location Binding	
	Registration	Static
Hosted SIP phone	Case-1	N/A
Single Registering PBX	Case-2	Case-5
Multi Registering PBX	Case-3	Case-6
Non-Registering PBX	Case-4	Case-7

The **Type of Enterprise SIP Endpoint** has the following values:

- A **Hosted SIP Phone** supports the registration procedures defined in [4].
- A **Single-Registering PBX** registers a single instance toward the service provider network (single contact address) for all of its subordinate users (e.g. [9]).
- A **Multi-Registering PBX** registers multiple instances toward the service provider network; one for each of its subordinate users.
- A **Non-Registering PBX** does not register with the service provider network.

The Type of Enterprise SIP Entity and the Network Name-Location binding essentially determine whether the ACS statically provisions data in the ESBC, or the ESBC dynamically learns and allocates the data itself at run time. For example, if the enterprise SIP entity is of a type that registers, then the ESBC can learn the entity's User contact address, whereas if the entity does not register then the ACS statically provisions the entity's User contact address information in the ESBC. Likewise, if Network Name-location binding is Registration, then the ESBC can allocate the Network contact address, whereas if it is Static then the Network contact address must be statically provisioned by the ACS.

There are also scaling considerations. For example, in the case of hosted IP-Centrex service, a single ESBC could serve hundreds of SIP phones. Therefore, it is desirable to minimize the ESBC provisioning overhead in this case by having the ESBC learn the SIP phone information and create as the phones register.

The individual use cases are described in detail in the following subsections. Since the ESBC is composed of one or more SBCs, the deployment cases are described in terms of SBC behavior.

I.1.1 Network Name-Location Binding conveyed via Registration

In this mode, the enterprise could contain SIP phones or SIP-PBXs. If the enterprise network contains SIP phones (i.e. hosted IP-Centrex service), the ESBC will support the SIP registration procedures defined in [4] on its Network interface. If the enterprise network contains SIP-PBXs (i.e., SIP Trunk service), then the ESBC will support the SIP registration procedures defined for SIP Trunks; namely, as defined in [9]. The SBC can either allocate the registering Network contact address itself (essentially means allocate the Network contact address port number), or it can let the ACS allocate the port number. All SBCs within a given ESBC must be configured the same way; either to support all SBC-allocated Network ports or all ACS-allocated Network ports.

Case-1) Hosted SIP Phone

The SBC learns / allocates all data associated with the enterprise SIP phones as they register.

In typical Case-1 deployments, the ESBC supports a SIP-aware NAT-traversal function (aka SIP-ALG), but otherwise does minimal SIP interworking. Digest authentication is end-to-end between SIP phone and service provider network.

Since the SBC has no pre-provisioned data associated with the SIP phones, it doesn't place any limits on who can register — it simply relays all registration requests to the service provider network. The assumption is that the service provider network has the knowledge / responsibility to provide access control.

Case-2) Single-Registering PBX

All data describing the SIP-PBX is pre-provisioned in the SBC, with the exception of the User and Network contact addresses, which are learned and allocated.

In typical Case-2 deployments, the Service Provider configures the ESBC to operate in a B2BUA mode in order to perform more complex SIP interworking procedures; e.g., more extensive SIP header manipulation. The ESBC may contain credentials and respond to digest challenges from the service provider network on behalf of SIP-PBX.

In this case, the ESBC provides access control and allows only those enterprise SIP entities that match its pre-provisioned data to register.

There are two potential models for authentication in this case. In one model, authenticate is hop-by-hop; the SBC will authenticate the SIP-PBX over the User interface, and the service provider network will authenticate the SBC over the Network interface. In the second model, authentication is end-to-end; the SBC transparently passes authentication challenges and challenge responses between the service provider network and SIP-PBX.

Case-3) Multi-Registering PBX

The SBC registers once with the service provider network on the Network interface on behalf of a SIP-PBX that supports multiple registrations on the User interface.

The SBC is pre-provisioned with all the data required to support the single Network registration (registering Public User Identity, Private User Identity, Digest credentials, etc). The SBC dynamically learns the data associated with each SIP-PBX extension, as the PBX registers for that extension. Each SIP-PBX extension is mapped to the single Network-side registration instance via E.164 mapping.

Case-4) Non-registering PBX

All data describing the SIP-PBX must be pre-provisioned in the SBC, including the User contact address of the PBX, and the data to support registration on the Network interface (The Network contact address can be pre-provisioned or allocated at runtime).

I.1.2 Network Name-Location Binding is Static

In this mode the enterprise network contains only SIP-PBXs (i.e., Static mode is only used for SIP trunking service). The Network contact address and the SIP-level Network routing information must be pre-provisioned in the SBC by the ACS. SIP messages exchanged between the service provider and enterprise network are routed based on the DNS routing procedures defined in [10].

Case-5) Single-Registering SIP-PBX

All data describing the SIP-PBX except the SIP-PBXs User contact address is pre provisioned in the SBC by the ACS.

Since there are only one or a few PBXs in this case, there is not much value-add in learning the User-side PBX identity information; it can be pre-provisioned.

Case-6) Multi-Registering PBX

All Network-side info must be pre-provisioned by the ACS. However, the User-side PBX extension info should be dynamically learned by the SBC as the PBX registers for each extension.

Case-7) Non-registering PBX

All data describing the SIP-PBX, including the PBX's User contact address, must be pre-provisioned.

If there are multiple non-registering PBXs, then in addition the SBC must be provisioned with data to route incoming SIP requests to the correct PBX. The actual form of this routing data is dependent on whether the service provider network or the ESBC does the E.164-to-PBX mapping.

If the E.164 mapping information is configured in the service provider network, then each PBX can be mapped to a separate SBC within the ESBC. The service provider network E.164 mapping table would then map requests sent to the enterprise to the Network address of the SBC associated with the target PBX.

If the E.164 mapping information is configured in the ESBC, then all PBXs are mapped to the same SBC. The service provider network then sends all requests destined for the enterprise to the same

place; to the Network address of the single SBC. The SBC then uses its E.164 mapping table to direct the request to the appropriate PBX.

I.2 Interwork Table Object Mode Parameters

The ACS provisions four mode parameters in each *Interwork* object that together identify the deployment use case for that *Interwork* instance.

The deployment use case is driven first by the type of service being provided – hosted IP Centrex or SIP trunking – and then, within SIP trunking service, by the capabilities / configuration of the SIP-PBX in the enterprise network and the configuration of the SIP Trunk to the service provider network. For example, the SIP-PBX may support different types of registration towards the ESG User interface or may not support registration at all. Likewise, the SIP Trunk may be configured for Registration mode or Static mode. The ESG must be capable of supporting all combinations of SIP-PBX types and SIP Trunk modes. To accommodate this, the VoiceService:2 data model defines two independent *Interwork* mode parameters; *NetworkConnectionMode* that describes the SIP interface towards the service provider network, and *UserConnectionMode* that describes the SIP interface towards the enterprise SIP Endpoint (for completeness, the values of these mode parameters are extended to cover hosted IP-Centrex service). The deployment use case is then defined by the value-combination of these two mode parameters.

The third *Interwork* mode parameter called *E164Routing* further defines the deployment use case by indicating whether or not the ESBC performs E.164 routing of incoming calls from the service provider network.

A fourth mode parameter called *NetworkAuthenticationChallengeMode* defines how the ESBC treats Digest authentication challenges received from the service provider network.

Here is a list of the four mode parameters and their values:

1) **NetworkConnectionMode:** describes the service being provided by the service provider network on the ESBC Network interface, and how the ESBC Network-facing objects are created to support that service (whether ACS-created or ESBC-created). Specifically, *NetworkConnectionMode* indicates; a) whether the service is hosted IP-Centrex or SIP trunking, b) for SIP trunking, whether or not the ESBC registers on its Network interface with the service provider network, and c) whether the Network-facing objects or parameters such as *SIP.Client and SIP.Client.{i}.Contact* are statically provisioned by the ACS or dynamically allocated by the ESBC.
   NetworkConnectionMode has the following values:

   a. **Static:** This mode applies only to SIP trunking service. The ESBC does not register with the service provider network. The *SIP.Client* and *SIP.Client.{i}.Contact* objects are created by the ACS.

   The remaining modes indicate that the ESBC conveys the enterprise SIP Endpoint contact information to the service provider network dynamically via SIP registration.

   b. **RegisterDynamic:** This mode applies only to SIP trunking service. The ESBC registers with the service provider network. The *SIP.Client* object is pre-provisioned

by the ACS. The ESBC allocates the client's Network contact port number dynamically at registration time.

c. **RegisterLearn:** This mode applies only to hosted IP-Centrex service. The SBC dynamically creates the *SIP.Client* and *SIP.Client.{i}.Contact* objects when the SIP phone registers. The purpose of this mode is to streamline the provisioning process for the hosted IP-Centrex case by taking advantage of the fact that the client information can be learned when the Endpoint registers.

d. **RegisterStatic:** This mode applies only to SIP trunking service. It is the same as *RegisterDynamic* except that the client's contact address is statically provisioned by the ACS.

2) **UserConnectionMode**: describes the SIP Endpoints in the enterprise network and how the ESBC User-facing objects are created to support those Endpoints. *UserConnectionMode* has the following values:

a. **Static:** This mode applies only to SIP trunking service. *SIP.Registrar.{i}.Account* and *SIP.Registrar.{i}.Account.{i}.Contact* objects are statically provisioned by the ACS. The SIP-PBX does not register.

The remaining modes indicate that the enterprise SIP Endpoint conveys its User contact information to the ESG dynamically via SIP registration.

b. **RegisterDynamic:** This mode applies only to SIP trunking service, and only for SIP-PBXs that register. The *SIP.Registrar.{i}.Account* object is provisioned by the ACS, but the account contact information is learned when the SIP-PBX registers. For multi-registering SIP-PBXs, the *Trunk.{i}.DDIRange* parameter is pre-provisioned by the ACS.

c. **RegisterLearn:** This mode applies to both the hosted IP-Centex and SIP trunking service. The *SIP.Registrar* instance is pre-provisioned by the ACS. The ESBC dynamically creates *SIP.Registrar.{i}.Account* objects as the enterprise Endpoints register. If the *NetworkConnectionMode* is also *RegisterLearn*, then it means that this is a hosted IP-Centrex case. If the *NetworkConnectionMode* is one of the SIP Trunk modes, then it means that this is a multi-registering SIP-PBX case where the ESBC learns the E.164 mapping info based on information received in the SIP-PBX registration transactions. In this SIP-PBX case, in addition to dynamically creating *SIP.Registrar.{i}.Account* instances, the ESBC dynamically updates the *Trunk{i}.DDIRange* parameter as it receives registration requests from the PBX. The purpose of this mode is to streamline the provisioning process for deployment scenarios where the ESBC receives many registrations on its User interface.

3) E164Mode: indicates whether or not the ESBC performs E.164 routing of incoming requests toward the SIP-PBX. E.164 routing is required for SIP Trunking scenarios when a single *SIP.Client* is mapped to multiple *SIP.Registrar.{i}.Account* objects. This can happen in two situations; when a single SIP Trunk is serving multiple Single-Registering SIP-PBXs, and when a single SIP Trunk is serving one or more Multi-Registering SIP-PBXs. E164Mode has two values:

a. true: The ESBC performs E.164 routing. Each *Trunk* object contains a *DDIRange* that indicates the set of E.164 numbers assigned to that Trunk.

b. false: The ESBC does not perform E.164 routing. The *Trunk* objects do not contain *DDIRange*.

4) ***NetworkAuthenticationChallengeMode*:** indicates how the ESBC treats digest authentication challenges received from the service provider network. Three values are defined:

a. ***PassThru***: the ESBC simply passes authentication challenges received from the service provider network through to the enterprise endpoint. Likewise, the ESBC passes authentication challenge responses received from the enterprise Endpoint through the service provider network. In this case the digest credentials (*AuthUserName*, *AuthPassword*) in the *SIP.Client* and *SIP.Registrar.{i}.Account* instances do not need to be pre-provisioned (and pre-provisioned credentials are ignored).

b. ***RespondLocal***: the ESBC responds to authentication challenges received from the service provider network on behalf of the enterprise Endpoint. The ESBC does not send authentication challenges to the enterprise Endpoint. In this case digest credentials are pre-provisioned in the *SIP.Client* object, since they're needed to generate the challenge response. The *SIP.Registrar.{i}.Account* object credentials do not need to be pre-provisioned (and pre-provisioned credentials are ignored).

c. ***Hop-by-Hop***: the ESBC responds to authentication challenges received from the service provider network on behalf of the enterprise Endpoint. In addition, the ESBC generates authentication challenges toward the enterprise Endpoint, and verifies the resulting challenge responses received from the enterprise Endpoint. In this case, digest credentials are pre-provisioned in both the *SIP.Client* and *SIP.Registrar.{i}.Account* instances.

I.3 ESBC Mode Parameter Examples

This section describes how the *NetworkConnectionMode, UserConnectionMode* and *E164Mode* parameters described in section I.2 support the use cases described in section I.1.

The *NetworkAuthenticationChallengeMode* parameter is not shown in the following use case examples, since its application is more-or-less orthogonal to the other three mode parameters. It should be noted however that there is some loose coupling between this authentication mode parameter and the other mode parameters. As shown in Table 5, some of the *NetworkAuthenticationChallengeMode* values require pre-provisioned digest credentials in the *SIP.Client* and/or *SIP.Registrar.{i}.Account* instances. However, digest credentials cannot be provisioned for client or account when they are created dynamically by the ESBC; i.e. when the *NetworkConnectionMode* (for *SIP.Client*) or *UserConnectionMode* (for *SIP.Registrar.{i}.Account*) is *RegisterLearn*. Therefore, the ACS will ensure that a compatible set of values is provisioned across all of these mode parameters.

Table 5 – Credentials Dependency for Digest Challenge Mode

NetworkAuthenticationChallengeMode	Requires Client Credentials	Requires Account Credentials
PassThru	No	No
RespondLocal	Yes	No
HopByHop	Yes	Yes

I.3.1 Network Name-Location Binding conveyed via Registration

Case-1) Hosted IP-Centrex Service

Description:

The enterprise network contains SIP phones that register individually. The ACS pre-provisions minimal data in the ESBC; the ESBC creates and maps a *SIP.Registrar.{i}.Account / SIP.Client* pair as each phone registers.

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterLearn	RegisterLearn	false

Initial Configuration:

For the hosted IP-Centrex case, the ACS initially provisions the objects / parameters shown in Figure 22. Once the ACS has created the *SIP.Network* and *SIP.Registrar* entries, it creates the *UserInterface* entry to link the two.

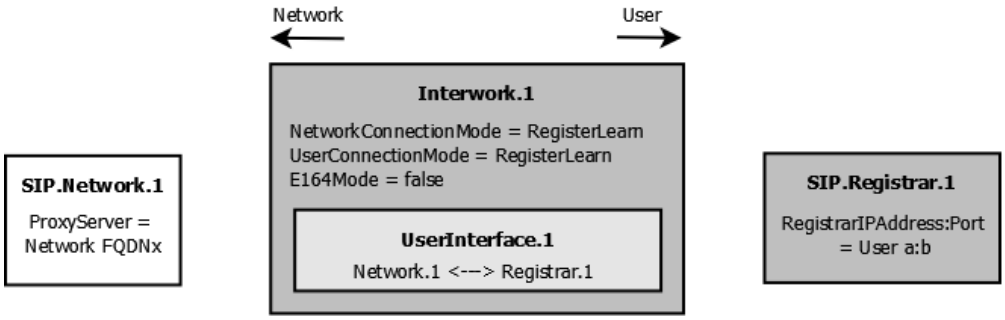

Figure 22 – Initial Configuration for Hosted IP-Centrex

Final Configuration:

The SIP phones in the enterprise network are provisioned to send SIP requests to the User listening *RegistrarIPAddressPort* "a:b" configured for *SIP.Registrar.1*. On receiving the initial REGISTER request from one of the SIP phones, the ESBC dynamically adds a *SIP.Registrar.1.Account* and *SIP.Client* object, and an *Interwork.1.Map* entry linking the two, as shown in Figure 23. The ESBC sets the URI and User contact addresses of the registering SIP phone in the account and client to

enable User / Network IP address interworking. The ESBC monitors the SIP messages, and will delete the dynamically added instances if this registration transaction fails or if this registration instance is explicitly terminated or expires.

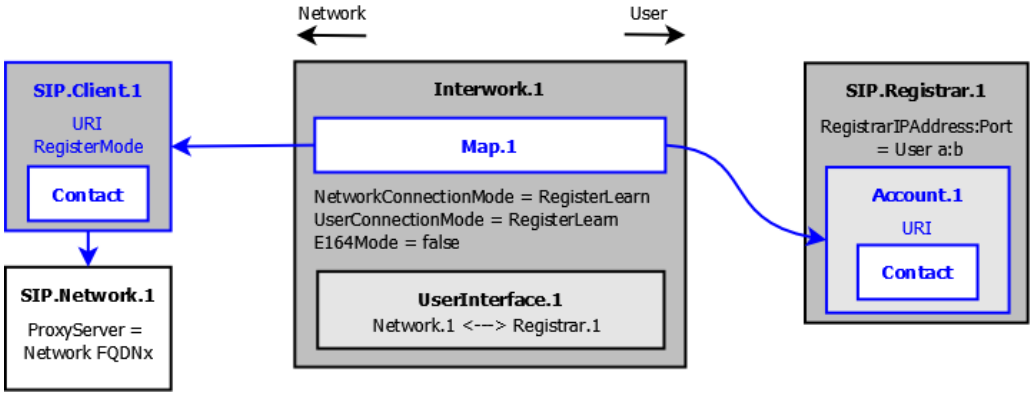

Figure 23 – Configuration After First REGISTER Request for Hosted IP-Centrex

The ESBC will continue to add *SIP.Registrar.1.Account / SIP.Client* pairs as additional SIP phones register.

Case-2) Register-Mode SIP Trunk to Single-Registering SIP-PBX

There are a number of sub-cases for the Single-Registering SIP-PBX, based on local policy such as whether the SIP-PBX's Network contact address is dynamically allocated or statically provisioned.

Case-2a) Reg-Mode SIP Trunk to one Single-Reg SIP-PBX, Dynamic Network Contact

Description:

This is the mainline SIP trunking use case, where the SIP-PBX supports a single SIP registration toward the network and the service provider SIP Trunk is configured for Registration Mode (i.e. the SIP Trunk supports the registration procedure in RFC6140). The ACS pre-provisions the ESBC *SIP.Registrar.{i}.Account* and *SIP.Client* instances with their URIs, so the ESBC can identify the registering SIP-PBX. The ACS may also pre-provision account and client credentials to support authentication. The ESBC dynamically sets the User and Network contact addresses when the SIP-PBX registers.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterDynamic	RegisterDynamic	false

Initial Configuration:

Figure 24 shows the initial ESBC configuration provisioned by the ACS for this use case.

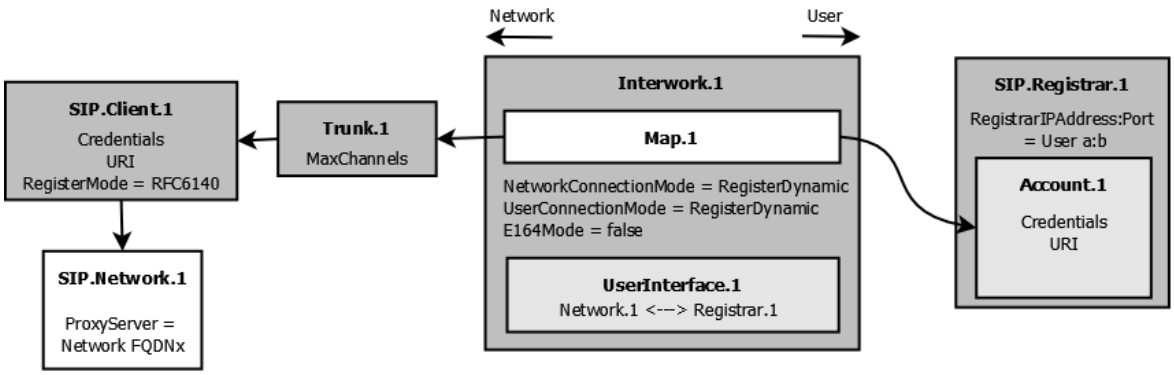

Figure 24 – Initial Configuration for Case-2a

Final Configuration:

On receiving the initial REGISTER request from the SIP-PBX, the ESBC sets the *SIP.Registrar.1.Account* User contact address based on the REGISTER Contact header field, and allocates a *SIP.Client* Network contact address, which is then populated in the Contact header field of the RFC6140 REGISTER request sent to the service provider.

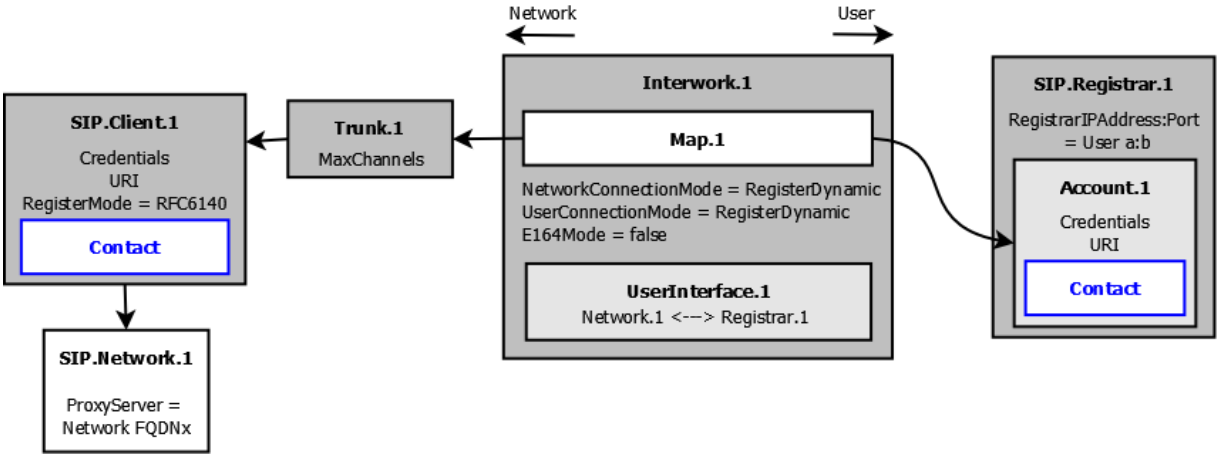

Figure 25 – Final Configuration for Case-2a

Case-2b) Reg-Mode SIP Trunk to one Single-Reg SIP-PBX, Static Network Contact

Description:

Same as Case-2a except that the Network contact address is statically provisioned by the ACS; i.e., the ESBC registers with the service provider network via the Network interface, but with a pre-provisioned contact address instead of a dynamically allocated address.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterStatic	RegisterDynamic	false

Case-2c) Reg-Mode SIP Trunk to multiple Single-Reg SIP-PBXs

Description:

Same as Case-2a except that the SIP Trunk is serving multiple Single-Registering SIP-PBXs. In other words, the ESBC receives registrations from multiple Single-Registering SIP-PBXs on its User interface, but sends only a single registration instance to the service provider network on its Network interface. The Network contact address could be dynamically allocated (shown here), or statically provisioned (*NetworkConnectionMode = RegisterStatic*).

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterDynamic	RegisterDynamic	true

Initial Configuration:

Figure 26 shows the initial ESBC configuration provisioned by the ACS for this use case. Multiple *SIP.Registrar.{i}.Account* objects are provisioned; one per registering SIP-PBX. Each *SIP.Registrar.{i}.Account* entry is mapped to a unique *Trunk* instance. Each *Trunk* instance is provisioned with the *DDIRange* of the associated SIP-PBX. The multiple *Trunk* instances are all linked to a single *SIP.Client* object.

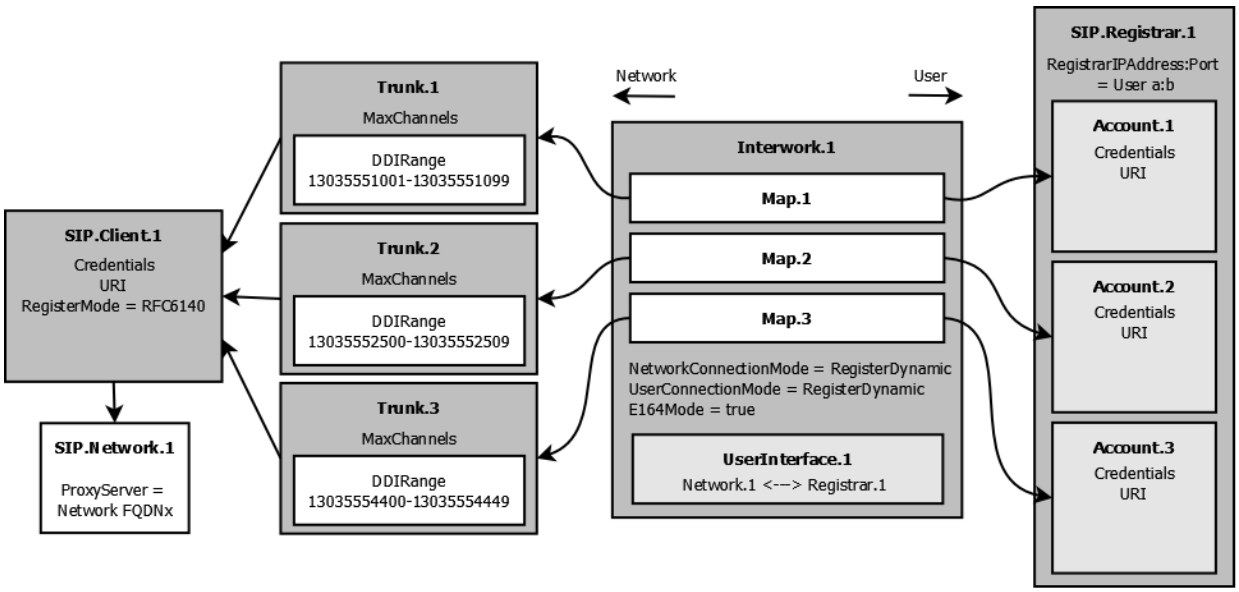

Figure 26 – Initial Configuration for Case-2c

Final Configuration:

On receiving the initial REGISTER request from a SIP-PBX, the ESBC finds the *Account* URI that matches the URI in the To header field of the received REGISTER request. The ESBC sets the

SIP.Registrar.1.Account User contact address based on the REGISTER Contact header field. The ESBC allocates a *SIP.Client* Network contact address, which is also populated in the Contact header field of the RFC6140 REGISTER request sent to the service provider. As each subsequent SIP-PBX registers, the ESBC updates the contact address of the target account as described above. As long as there is at least one registered account, the ESG will maintain the client registration. If the number of registered *SIP.Registrar.1.Account* instances drops to zero, then the ESBC will de-register the client registration.

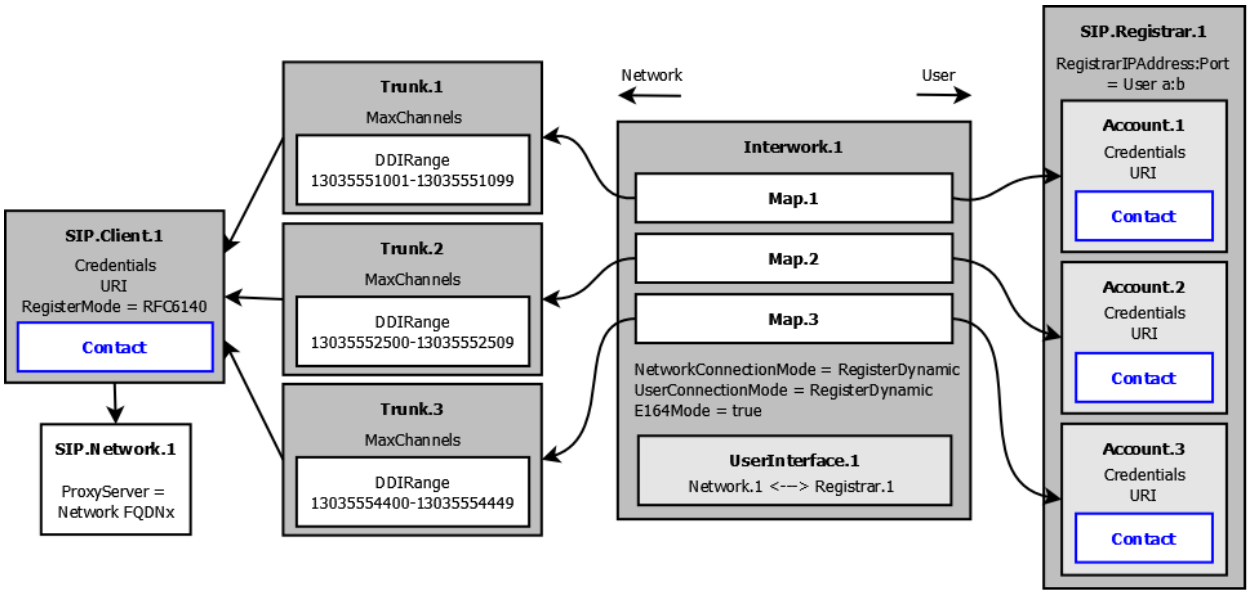

Figure 27 – Final Configuration for Case-2c

Case-3) Multi-Registering SIP-PBX

As in the Single-Registering SIP-PBX case, there are multiple sub-cases when the SIP-PBX supports multiple registrations toward the network.

Case-3a) Reg-Mode SIP Trunk to one Multi-Reg SIP-PBX, Dynamic Network Contact

Description:

The service provider SIP Trunk is configured for Registration Mode. Therefore, the ESBC must honor the multiple registrations received from the SIP-PBX, but generate a single registration toward the service provider network.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterDynamic	RegisterLearn	true

Initial Configuration:

The ACS pre-provisions the objects and parameters shown in Figure 28. Once the objects are created, the ESBC listens for incoming REGISTER requests on the configured *SIP.Registrar.1 .RegistrarIPAddress:RegistrarPort* (i.e. the registrar's listening User IP address:port).

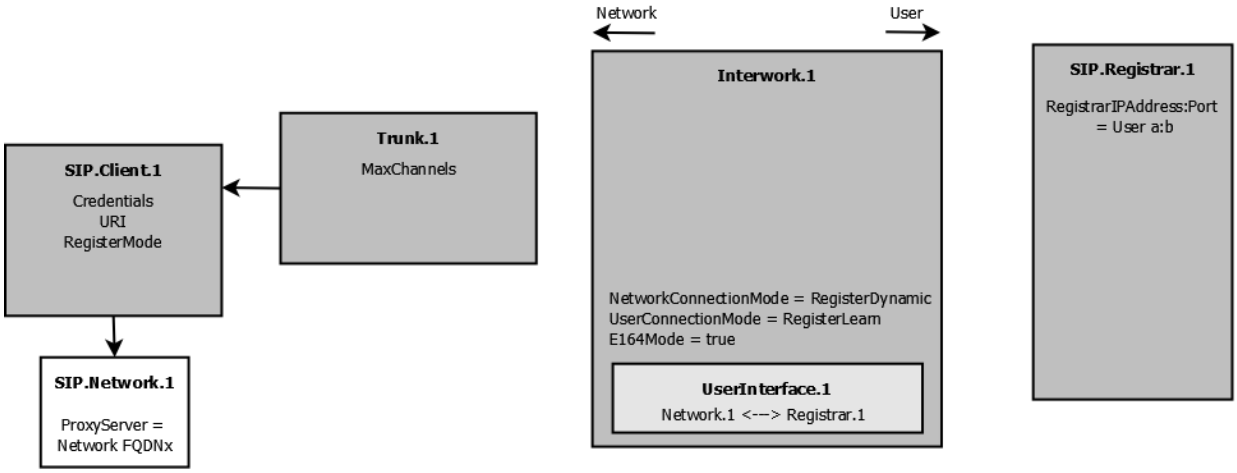

Figure 28 – Initial ESBC Configuration for Case-3a

Final Configuration:

When the ESBC receives the first REGISTER request from the SIP-PBX on the User listening IP address:port of *SIP.Registrar.1*, it creates an *Account* instance and links it via an *Interwork.{i}.Map* object to the pre-configured *Trunk.1* object. The ESBC also adds the telephone number from the To header field of the REGISTER request to the *Trunk.1.DDIRange* parameter. Since this is the first REGISTER request from the SIP-PBX, the ESBC allocates a Network contact IP address:port for the client and initiates a registration procedure toward the service provider network.

As it receives each subsequent REGISTER request from the SIP-PBX, the ESBC allocates an additional *SIP.Registrar.1.Account* instance, maps it to the single pre-provisioned Trunk and updates the *Trunk.1.DDIRange* to include the new telephone number of the associated registering SIP-PBX line. The ESBC does not allocate a Network contact address for the client for these subsequent REGISTER requests, since one has already been allocated. In the example shown in Figure 29, the ESBC has received three REGISTER requests from the SIP-PBX, and in turn has created three *SIP.Registrar.1.Account* instances and a *Trunk.1.DDIRange* identifying three telephone numbers.

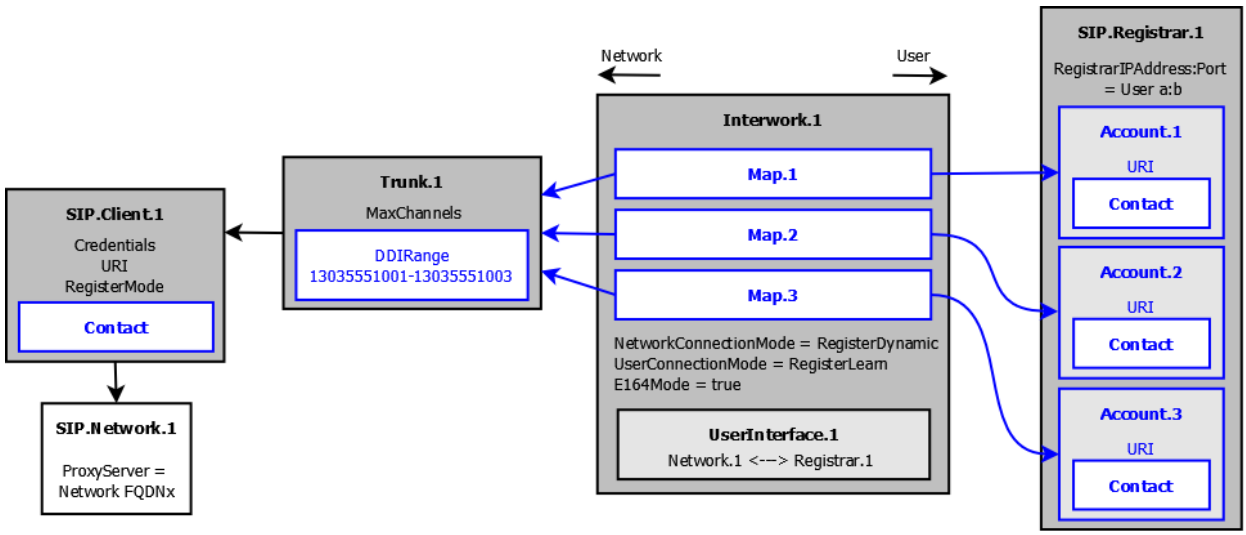

Figure 29 – Final ESBC Configuration for Case-3a

Case-3b) Reg-Mode SIP Trunk to one Multi-Reg SIP-PBX, Static Network Contact

Description:
Same as Case-3a except that the Network contact address is statically provisioned by the ACS; i.e., the ESBC registers with the service provider network via the Network interface, but with a pre-provisioned contact address instead of a dynamically allocated address.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterStatic	RegisterLearn	true

Case-3c) Reg-Mode SIP Trunk to multiple Multi-Reg SIP-PBXs

Description:
Same as Case-3a except that the SIP Trunk is serving multiple Multi-Registering SIP-PBXs. The Network contact address could be dynamically allocated (shown here), or statically provisioned (*NetworkConnectionMode = RegisterStatic*).

Mode Parameters:

NetworkConnectionMode	ConnectionMode	E164Mode
RegisterDynamic	RegisterLearn	true

Initial Configuration:

As shown in Figure 36, this case can be supported with the same initial configuration (including the same *Interwork* mode parameters) that is used for the single Multi-Registering SIP-PBX scenario

shown in Case-3a. In this example, the multiple SIP-PBXs in the enterprise network would be pre-configured to send their REGISTER requests to the ESBC listening User IP address:port of *SIP.Registrar.1*.

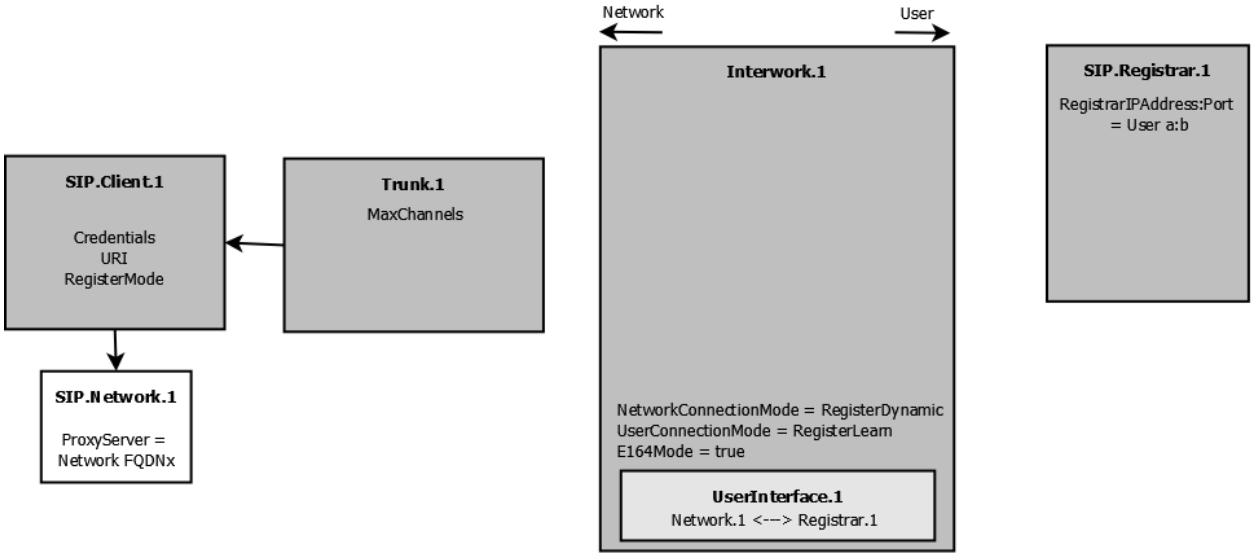

Figure 30 – Initial Configuration for Case-3c

Final Configuration:

Figure 31 shows the final configuration for a deployment example where there are two Multi-Registering SIP-PBXs; one with three lines, and one with two lines. When the ESBC receives the first REGISTER request from one of the SIP-PBXs on the User listening IP address:port of *SIP.Registrar.1*, it performs the same procedure described for Case-3a; i.e., it creates an *SIP.Registrar.1.Account* instance, links it to the pre-configured *Trunk.1* object, updates the *Trunk.1.DDIRange* parameter with the registering E164 number, and initiates a *SIP.Client* registration procedure toward the service provider network.

As it receives each subsequent REGISTER request from the SIP-PBX, the ESBC allocates an additional *SIP.Registrar.1.Account* instance, maps it to the single pre-provisioned *Trunk* instance and updates the *Trunk.1.DDIRange* parameter. In the example shown in Figure 31, the ESBC has received three REGISTER requests from one SIP-PBX and two from the other, which creates a total of five accounts and a Trunk's *DDIRange* identifying five telephone numbers.

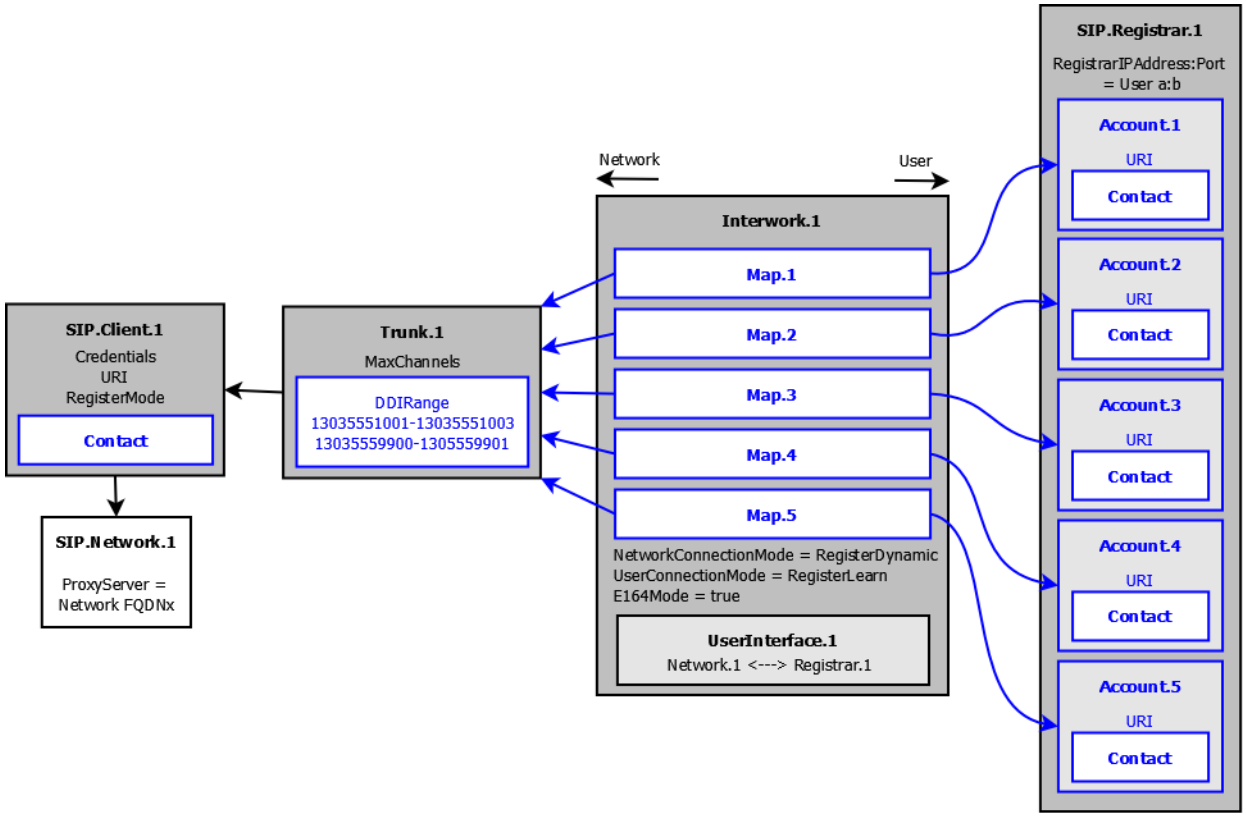

Figure 31 – Final Configuration for Case-3c

Alternate Case-3c Configuration:

In the above example, both of the Multi-Registering SIP-PBXs are served by the same *Trunk* object, which means that the parameter values for that Trunk, such as the maximum number of channels allowed, would be applied across both PBXs. There is an alternate case where a service provider wants to configure a different max-channel limit per PBX. This could be accomplished with the initial configuration shown in Figure 32. The ACS pre-provisions an empty *SIP.Registrar.{i}.Account* instance in each *SIP.Registrar*, so that each *Trunk* can be associated with its own *SIP.Registrar*. Each Multi-Registering SIP-PBX is pre-configured with a different next-hop address; one PBX with a next-hop pointing to the listening port of *SIP.Registrar.1*, and the other PBX with a next-hop pointing to the listening port of *SIP.Registrar.2*. Since each *SIP.Registrar* is linked to a different *Trunk* (via the pre-configured *SIP.Registrar.{i}.Account* instance), this arrangement enables each PBX to be associated with a different *Trunk* object.

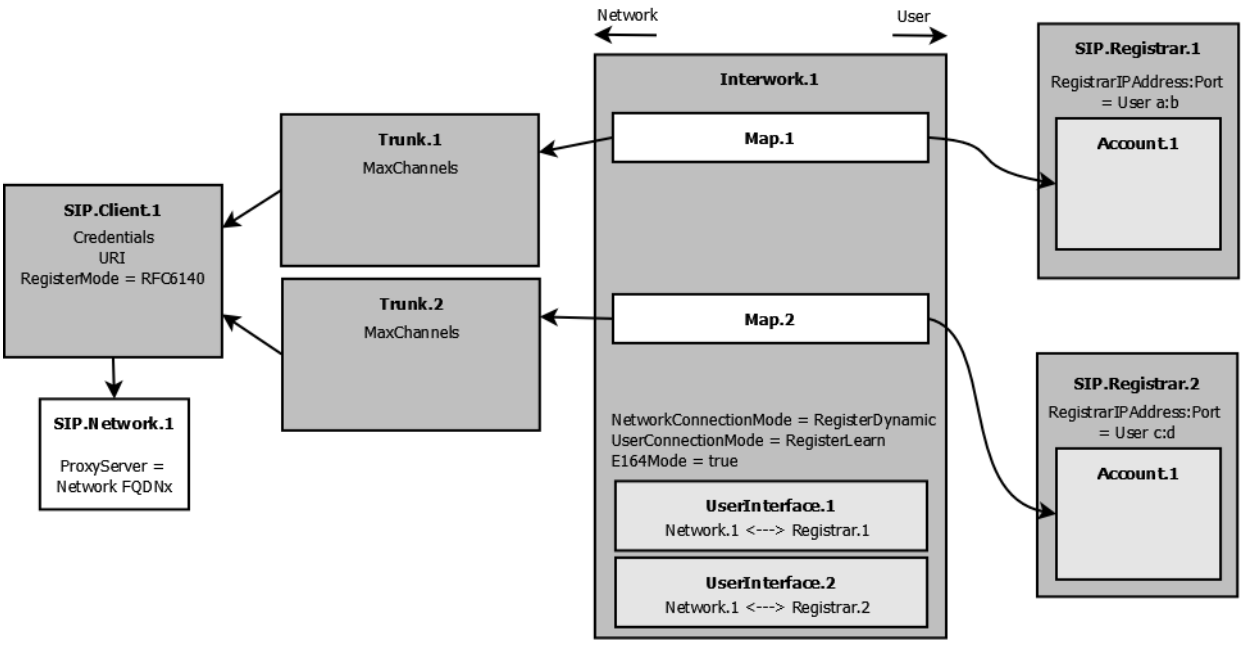

Figure 32 – Alternate Initial Configuration for Case-3c

Figure 33 shows the final ESBC configuration for this Case-3c alternative after both Multi-Registering SIP-PBXs have registered (three registrations from one PBX, two from the other).

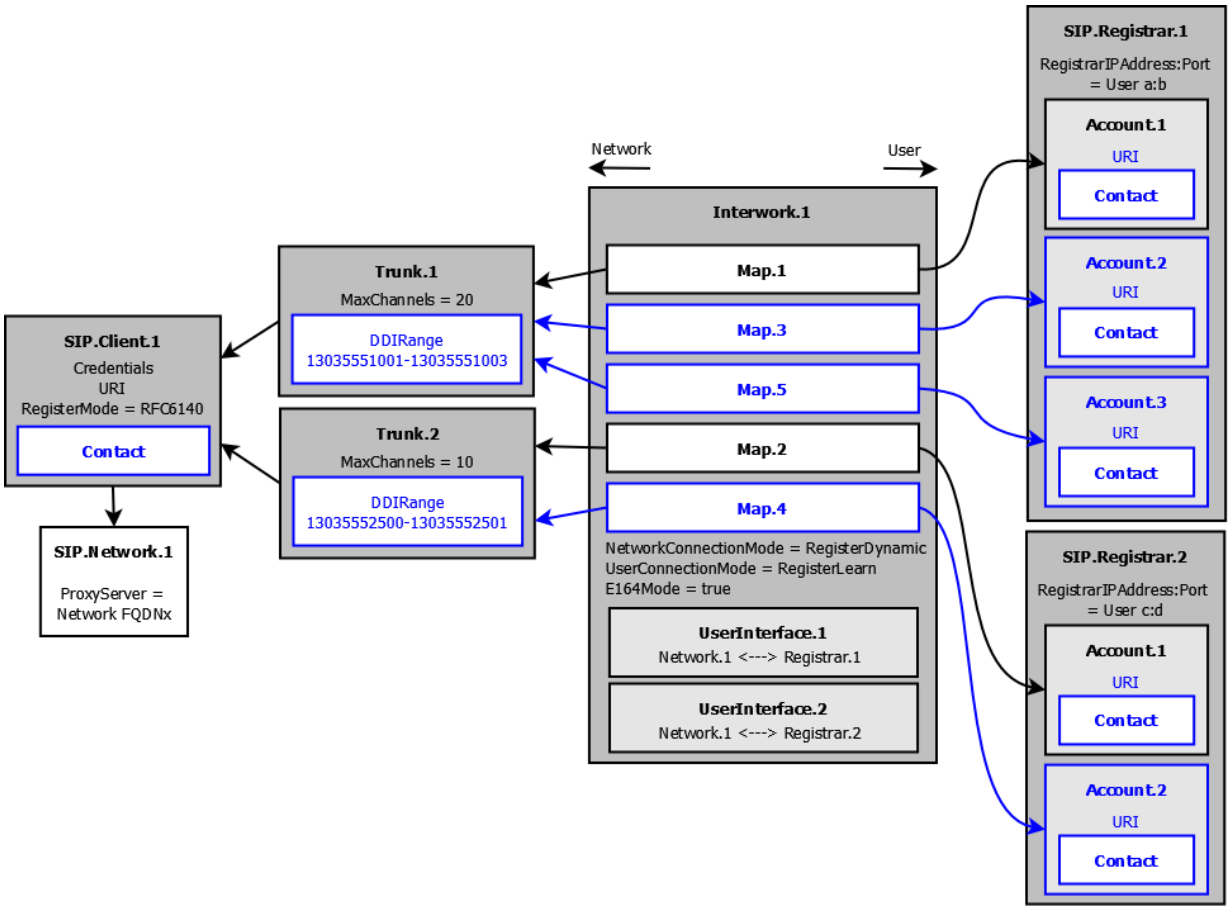

Figure 33 – Alternate Final Configuration for Case-3c

Case-4) Registration-Mode SIP Trunk to Non-Registering SIP-PBX

Case-4a) Reg-Mode SIP Trunk to one Non-Registering SIP-PBX, Dynamic Network Contact

Description:

The SIP-PBX doesn't register toward the ESBC User interface, while the ESBC does register toward the service provider network on its Network interface.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterDynamic	Static	false

Initial Configuration:

The ACS pre-provisions the objects and parameters shown in Figure 34.

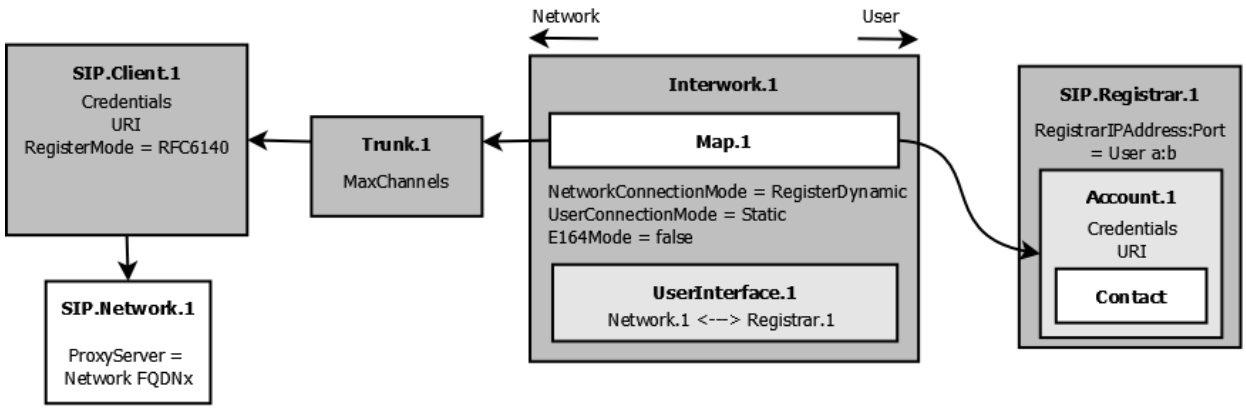

Figure 34 – Initial ESBC Configuration for Case-4a

Final Configuration:

When the ESBC detects that the non-registering SIP-PBX is available (say using a SIP OPTIONS ping), it allocates a client Network contact address as shown in Figure 35, and registers with the service provider network.

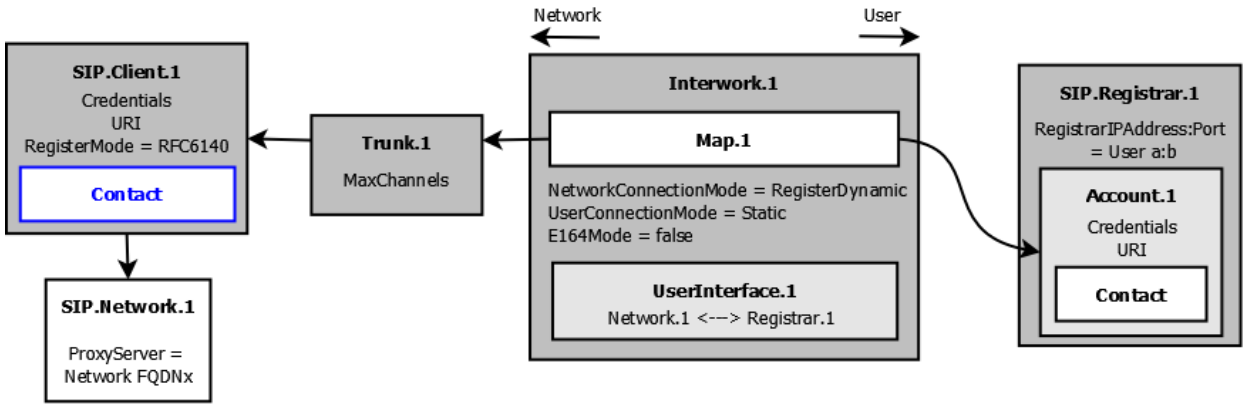

Figure 35 – Final ESBC Configuration for Case-4a

Case-4b) Reg-Mode SIP Trunk to one Non-Registering SIP-PBX, Static Network Contact

Description:

This is the same as Case-4a except that the Network contact address is statically provisioned; i.e., the ACS pre-provisions the client contact address.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterStatic	Static	false

Case-4c) Reg-Mode SIP Trunk to multiple Non-Registering SIP-PBXs

Description:

This is the same as Case-2c except that the Network contact address is statically provisioned; i.e., the ACS pre-provisions the client contact address. The Network contact address could be dynamically allocated (shown here), or statically provisioned (*NetworkConnectionMode* = *RegisterStatic*).

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
RegisterStatic	Static	true

The initial and final configurations are the same as shown for Case-2c, except that the account contact address is statically pre-provisioned by the ACS (and of course, the *UserConnectionMode* is *Static*).

I.3.2 Network Name-Location Binding is Static

This section describes the ESBC data model for the deployment use cases where the SIP Trunk to the service provider network is configured for static mode. There are fewer individual cases here than described in Appendix I.3, since in this category the ESBC Network contact address is always statically provisioned and service is provided only to SIP-PBXs (no SIP phones).

Case-5) Static Mode SIP Trunk to Single-Registering SIP-PBX

Case-5a) Static Mode SIP Trunk to one Single-Registering SIP-PBX

Description:

The SIP-PBX supports a single SIP registration toward the ESG User interface.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
Static	RegisterDynamic	false

Initial Configuration:

Figure 36 shows the initial ESBC configuration provisioned by the ACS for this use case. With the exception of the account contact address, all objects and parameters are pre-configured by the ACS.

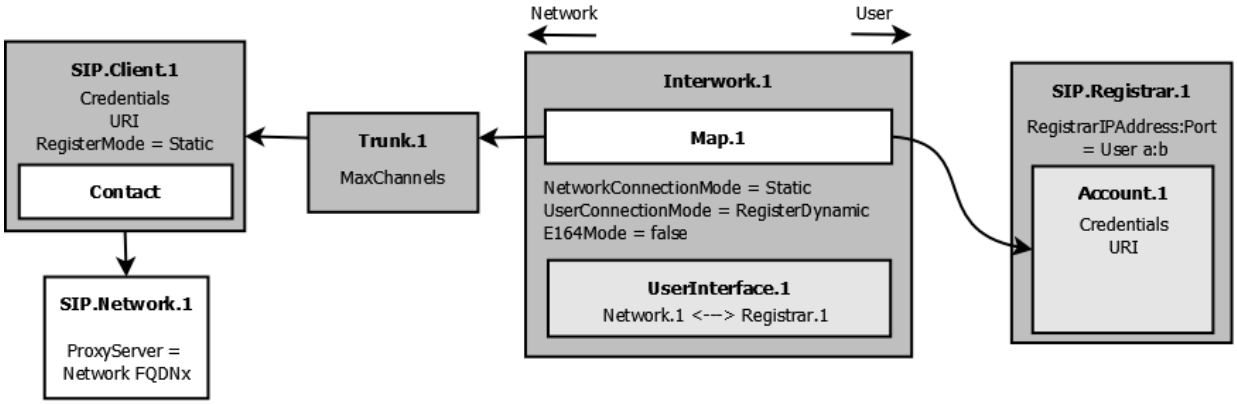

Figure 36 – Initial ESBC Configuration for Case-5a

Final Configuration:

When the SIP-PBX registers, the ESBC updates the account contact address based on the contents of the Contact header field in the received REGISTER request, as shown in Figure 37.

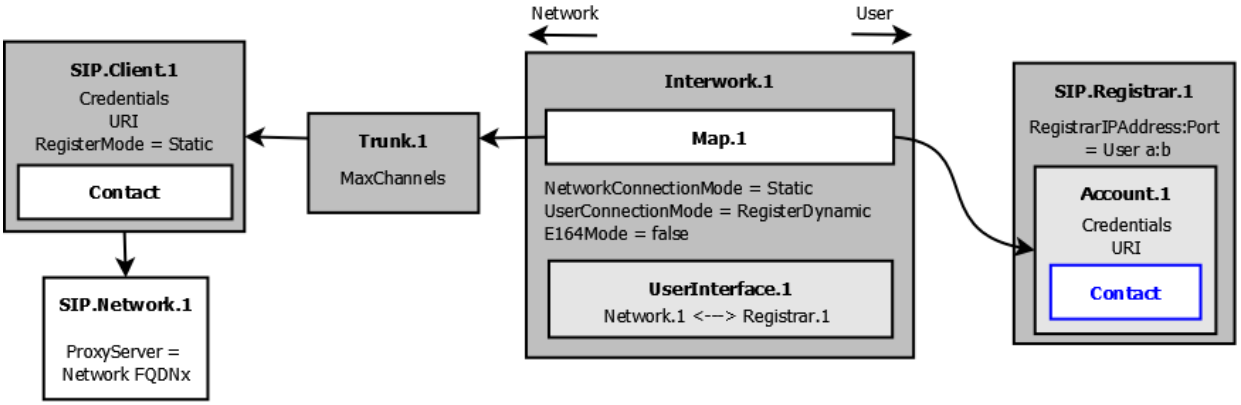

Figure 37 – Final ESBC Configuration for Case-5a

Case-5b) Static Mode SIP Trunk to multiple Single-Registering SIP-PBX

Description:

The enterprise network contains multiple SIP-PBXs, each PBX supporting a single SIP registration toward the ESG User interface. The ESBC provides a single statically provisioned SIP Trunk interface to the service provider network.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
Static	RegisterDynamic	true

As in Case-5a, the client contact address is statically provisioned, and the *Client.{i}.RegisterMode* = *Static*. Otherwise, this case is similar to Case-2c, where multiple *Trunk* objects are mapped one-to-one to multiple *SIP.Registar.1.Account* instances and each *Trunk* is pre-provisioned with a *DDIRange* that is used to route incoming calls to the correct SIP-PBX. Figure 38 shows the final configuration for this case.

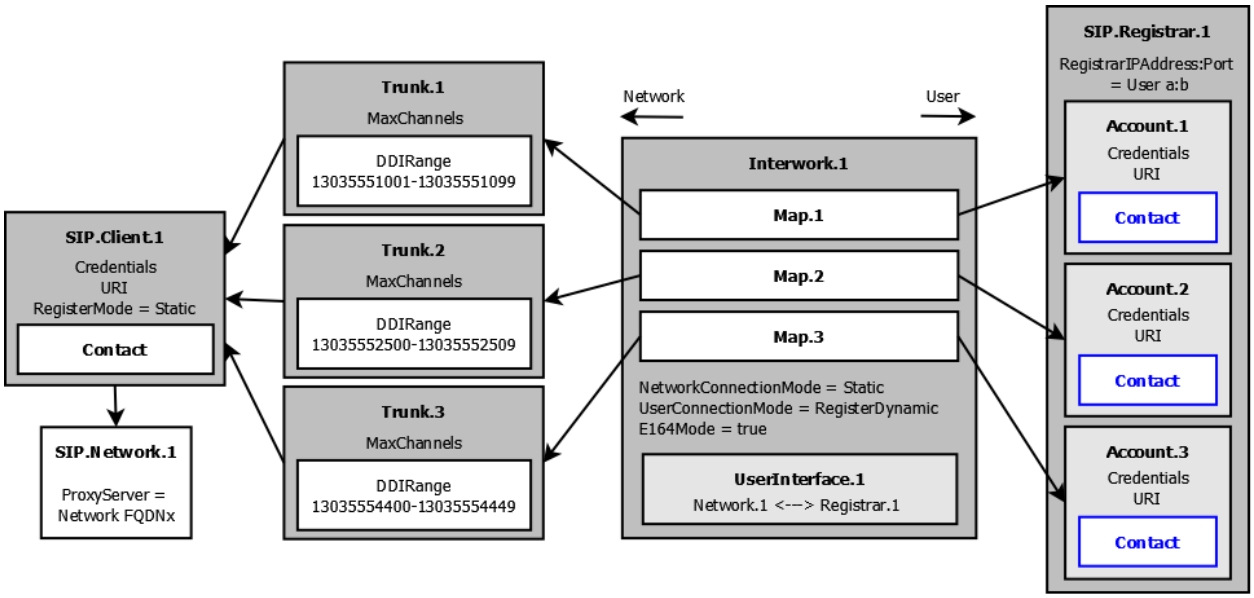

Figure 38 – Final ESBC Configuration for Case-5b

Case-1) Static-Mode SIP Trunk to Multi-Registering SIP-PBX

Description:

The SIP-PBX supports a multiple SIP registrations toward the ESG User interface, while the ESBC provides a single statically provisioned SIP Trunk interface to the service provider network.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
Static	RegisterLearn	true

Initial Configuration:

This case is the similar to Case-3b (for one PBX) or Case-3c (for multiple PBXs) except that the ESBC does not register toward the service provider network on its Network interface. The initial configuration pre-provisioned by the ACS is shown in Figure 39.

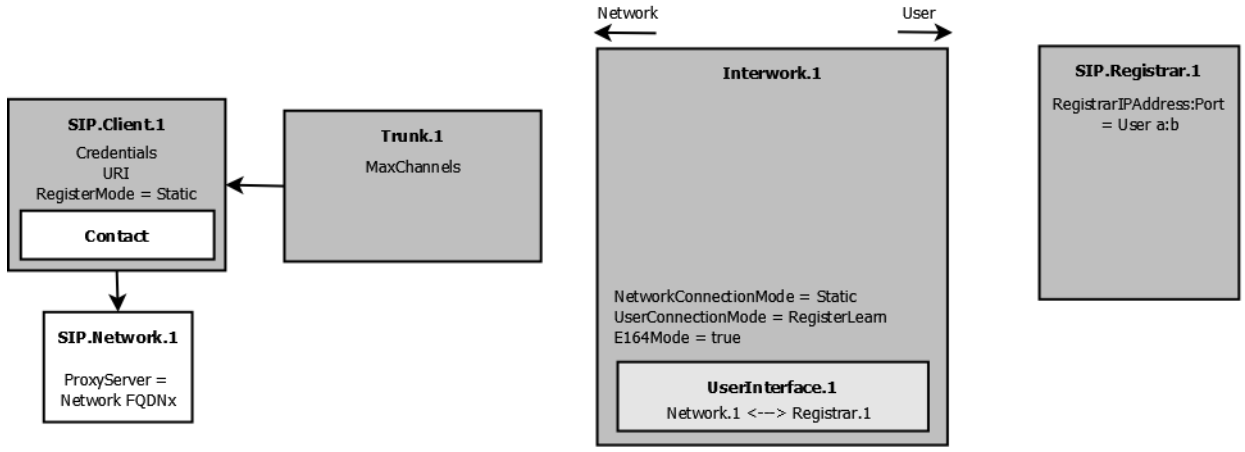

Figure 39 – Initial ESBC Configuration for Case-6

Final Configuration:

As the ESBC receives registration requests from the SIP-PBX, it creates *SIP.Registrar.1.Account* instances, links them via *Interwork.1.Map* table to the pre-provisioned *Trunk* instance and updates the Trunk's *DDIRange* parameter as described for Case-3a, in order to create the final configuration shown in Figure 40.

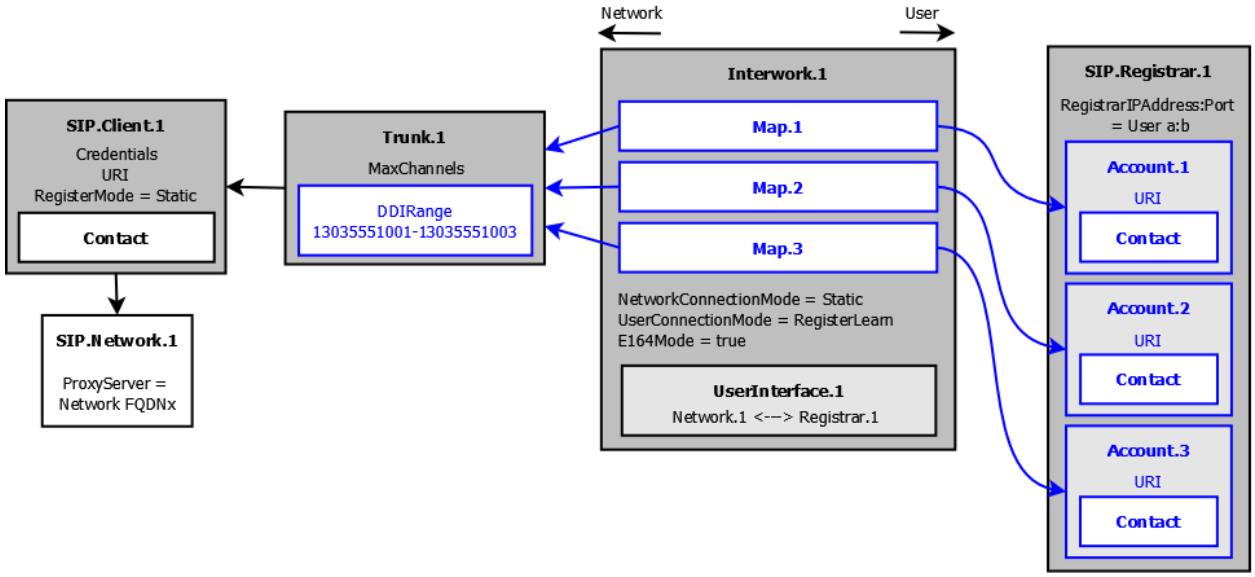

Figure 40 – Final ESBC Configuration for Case-6

Case-3c shows a couple of alternative ESBC configurations for support of multiple Multi-Registering SIP-PBXs. These same alternatives (along with the changes described here) would apply for this case.

Case-2) Static-Mode SIP Trunk to Non-Registering SIP-PBX

Case-7a) Static-Mode SIP Trunk to one Non-Registering SIP-PBX

Description:

The enterprise network contains a single Non-Registering SIP-PBXs.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
Static	Static	false

Initial Configuration:

Figure 41 shows the initial ESBC configuration pre-provisioned by the ACS for this use case. Multiple *SIP.Registrar.1.Account* instances are provisioned; one per registering SIP-PBX. All parameters in the *SIP.Registrar.1.Account* instances are also pre-provisioned, including the User contact address. Each account is mapped to a unique *Trunk* instance. Each *Trunk* instance is provisioned with the *DDIRange* parameter containing the associated SIP-PBX. The multiple *Trunk* instances are all linked to a single *SIP.Client* instance.

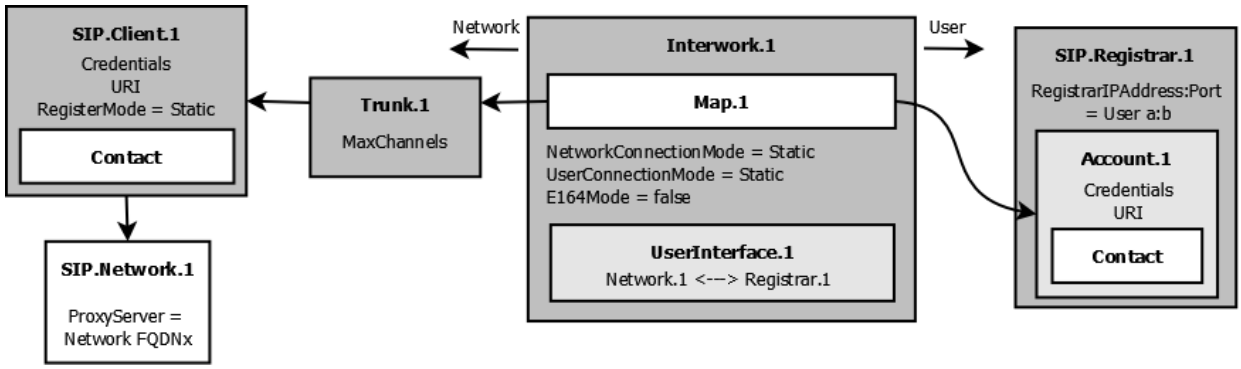

Figure 41 – Initial ESBC Configuration for Case-7a

Final Configuration:

Since all of the ESBC data is pre-provisioned for this case, the final configuration matches the initial configuration.

Case-7b) Static-Mode SIP Trunk to multiple Non-Registering SIP-PBXs

Description:

The enterprise network contains multiple Non-Registering SIP-PBXs.

Mode Parameters:

NetworkConnectionMode	UserConnectionMode	E164Mode
Static	Static	true

Initial Configuration:

Figure 42 shows the initial ESBC configuration pre-provisioned by the ACS for this use case. Multiple *SIP.Registrar.1.Account* instances are provisioned; one per registering SIP-PBX. All parameters in the *SIP.Registrar.1.Account* instances are pre-provisioned, including the User contact address. Each account is mapped to a unique *Trunk* instance. Each Trunk object is provisioned with the *DDIRange* of the associated SIP-PBX. The multiple *Trunk* objects are all linked to a single *Client* object.

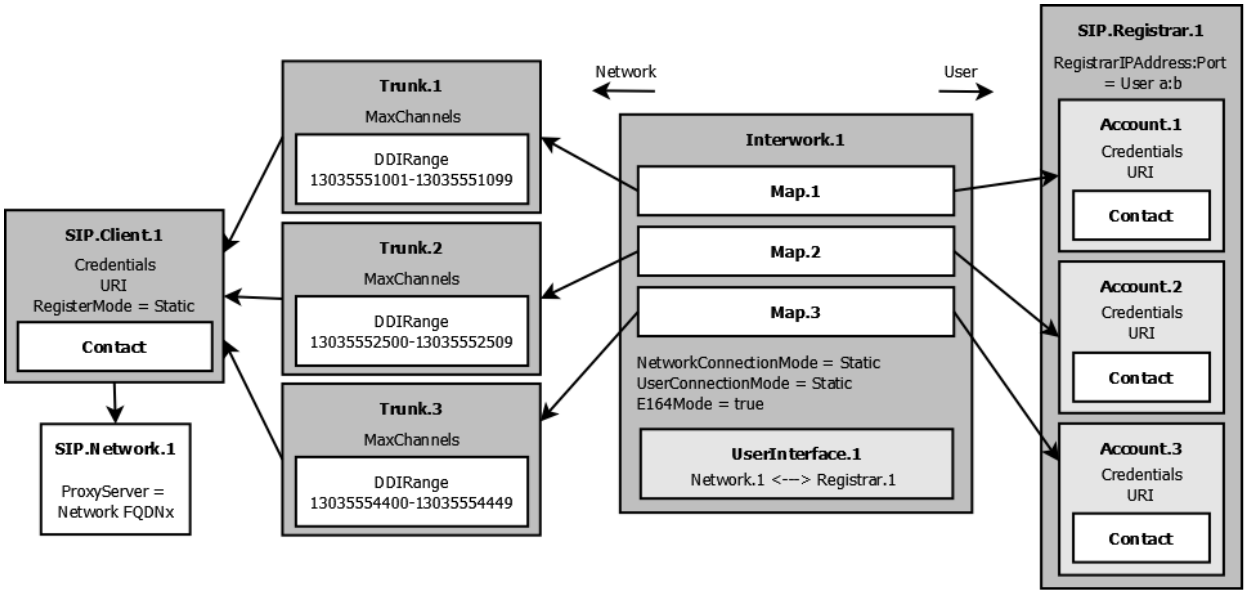

Figure 42 – Initial ESBC Configuration for Case-7b

Final Configuration:

Since all of the ESBC data is pre-provisioned for this case, the final configuration matches the initial configuration.

Appendix II. Clarifications Regarding Statistics

Most of the statistics parameters in the data model are self-explanatory counters. Some of them, however, require a solid definition to avoid discrepancies in the calculation algorithms used.

II.1 Overruns and Underruns

Those dejittering-related parameters are found in Line, Extension and Session DSP Stats objects. They are computed as follows:

- An overflow is the situation where the buffer receives new data but cannot insert it because it is full. Each period of time from the moment this situation occurs to the moment the situation returns to normal counts as one Overrun.
- An underflow is the situation where the buffer is required to provide data but cannot because it is empty. Each period of time from the moment this situation occurs to the moment the situation returns to normal counts as one Underrun.

II.2 CallLog Session statistics

II.2.1 Source and Destination

Those statistics are Source and Destination statistics. Depending on the Voice Service configuration and the type of call, the CPE has either Source, or Destination, or both. All counters for a missing side are to use the unknown value defined for the counter used.

II.2.2 BurstCount

RTP statistics use the concept of Burst, which is defined in [11] as a sequence of lost packets, possibly including islands of received packets. Each island consists of up to Gmin – 1 received packets (a sequence of Gmin received packets terminates the Burst and is not an island).
Gmin is writable, so that the count of Bursts can be configured as best suits the operator.

For example, if Gmin is 1 then there can be no islands and any sequence of lost packets is a Burst. If Gmin is 2 then some Bursts contain islands of 1 received packet. Each Burst is terminated by 2 consecutive received packets.
It is important to note that this definition considers single packet losses as Bursts of length 1.

II.2.3 PacketDelayVariation

According to [12] II.3 Definition of 1-point packet delay variation, the following calculation scheme is recommended:

1) Start calculation at the reception of the first packet of the first block of a call and record the local reception time of the first packet as E(0). Also initialize PDV_max and PDV_min to 0 at the start of the call.
2) For each received packet calculate the packet delay variation as PDV = (E(i) – E(0)) – Tacc. Tacc is the accumulated RTP packet time stamp difference of the first received packet and the ith received packet.

3) Record the maximum and the minimum of PDV (PDV_max and PDV_min).
4) After 5 minutes reset Tacc to Tacc=0 and E(0) to E(0) = local time of last received packet.
5) Continue with step 2.

As a result after the call, provide PDV_delta = PDV_max-PDV_min as the resulting TR-104 parameter.

The reset of Tacc and E(0) every 5 minutes is done to minimize the effect of clock drift between the sending and the receiving side on the result.

The value Tacc and, depending on the realization also the value of E, should be processed in units of RTP timestamps. Care has to be taken to properly handle the wraparounds of the 16 bit RTP timestamp. The resulting PDV_delta is expressed in milliseconds.

End of Broadband Forum Technical Report TR-104