

Media Services

Feature Description

Table of Contents

1 Introduction **1**

2 Architecture **2**
2.1 Call Processors and Media Processing Instances 2
2.2 Bringing a Call to the Media Server 2

3 Event-Driven Service Logic **2**
3.1 Events 2
3.2 Call State Across Events 3
3.3 Rules for Media Service Logic 4

4 Media Nodes **4**
4.1 Answering and Releasing the Call 4
4.2 Playing Media 4
4.3 Collecting DTMF 5
4.4 Timers 5
4.5 Setting Up and Bridging Calls 5
4.6 Interacting with Other Calls 6
4.7 Speech-to-Text 6

5 Typical Services **6**

6 Example: Simple IVR **6**

1 Introduction

With media services, SRE does more than take routing decisions: it answers calls and controls them. A call is answered by a media server under SRE control, and a service logic decides what to do with it: play announcements, collect DTMF digits, set up new calls, bridge parties together or release the call. The SRE Product Description explains where media services fit in the platform.

This document describes how media services work: the architecture, the event-driven execution of the service logic, the nodes available, licensing, typical services and an IVR example. Text-to-speech and speech-to-text are mentioned briefly here and covered in detail in the SRE and AI feature description. The Media Guide explains step by step how to build and activate a media service logic.

2 Architecture

2.1 Call Processors and Media Processing Instances

Media services add Media Processing instances (MPs) to the SRE platform, next to the Call Processors. Each MP combines two layers:

- **A media server**, which terminates the SIP signalling and the RTP media streams towards the voice network.
- **The media processor**, which listens to the events reported by the media server, runs the media service logic and sends commands back to the media server.

The media processor receives events from the media server over a WebSocket and sends its commands through the media server's REST interface. Like the Call Processors, each MP holds a replicated copy of the database, so the media service logic can use the same data as the routing service logic. MPs keep handling calls if the connection with the Element Managers is lost. All MPs are active and work independently; capacity and redundancy are obtained by deploying several of them.

2.2 Bringing a Call to the Media Server

A call reaches a media server through a regular SIP service logic on a Call Processor. The **Relay to media server** node sends the call to one of the declared media servers:

- SRE monitors the media servers like other SIP agents, and selects one at random among those that are available.
- Selected variables of the SIP service logic can be passed to the media server. A routing decision taken on the Call Processor, such as the customer or the language, can therefore be reused in the media service.

3 Event-Driven Service Logic

3.1 Events

A SIP service logic usually runs once per call. A media service logic runs many times during the same call: once for each event reported for that call. The name of the event is available in the `event` variable, and a media service logic typically starts with a **Sequential conditions** node that branches on it.

Event	When it occurs	Additional data
start	A new call is received by the media processor	calling number and the variables passed by the Call Processor
dtmf	A DTMF digit is received	the digit (0-9, *, #, A-D)
playback_started	An announcement has started playing	the media being played
playback_finished	An announcement has finished playing	the media that finished
originated	A call set up with **Originate call** is answered	
originate_failed	A call set up with **Originate call** fails because the destination is busy or congested	the failure cause
timeout	A timer set with **Set timeout** expires	
stt_finished	A speech-to-text recording has been transcribed	the transcribed text
custom events	An event is sent to the call with **Broadcast event**	the event name

Each execution handles one event, performs the requested actions and ends. The call then waits for the next event.

3.2 Call State Across Events

Because a call is handled through several executions, the variables are saved at the end of each execution and restored before the next one for the same call. A value set when the call starts, such as the customer identifier or the digits collected so far, is still available when the next DTMF digit or the end of an announcement is reported.

When the caller hangs up, SRE releases the other call legs and the bridge associated with the call, and removes the call state shortly afterwards. A call state that exceeds the configured maximum lifetime is also removed, and its call legs are released.

3.3 Rules for Media Service Logic

All the general-purpose nodes of the Service Logic Editor are available in a media service logic: conditions, variables, database queries, HTTP requests, sub-logics and so on. The following rules apply:

- The nodes of the SIP and Accounting groups cannot be used.
- The service logic must end with an output node of the Media group or with an **Exit service logic** node.

A media service logic can be tested with the simulation tools, using the *Simulate Media* tab to generate start, DTMF, playback finished, originated and timeout events. Call tracing shows the events received for a live call and the nodes executed for each of them.

4 Media Nodes

The Media group contains two kinds of nodes. Processing nodes perform an action and continue to the next node, so several of them can be chained in one execution. Output nodes end the execution with a final action. Several output nodes can still play a role in the same call, because each event leads to a new execution.

4.1 Answering and Releasing the Call

- **Answer** answers the incoming call.
- **Hangup call** releases the call. For a call that has not yet been answered, the SIP release cause can be selected, for example Busy (486), No answer (480), Rejected (403), Unallocated (404) or Normal (603).

4.2 Playing Media

- **Play media file** plays an audio file stored on the media server. The standard set of prompts is proposed in the node, and the file name can also come from a variable. A language can be selected to use the corresponding version of the prompt.
- **Play text-to-speech** generates audio from a text, which can be built from variables, and plays it. Generated prompts are cached on the media server and reused. Providers, languages, voices and SSML are described in the SRE and AI feature description.
- **Play music on hold** and **Stop music on hold** start and stop music on hold, with the default class or a specific class configured on the media servers.

- **Stop media** stops the announcements being played on the call, for example when the caller starts typing digits.

The two play nodes have a direction: to the caller, to the other party of the call, or to both parties once they are bridged. Several announcements can be played at the same time, for example music on hold together with a periodic “please hold” message.

4.3 Collecting DTMF

Each digit pressed by the caller triggers a dtmf event. The service logic reacts to the digit directly, as in a menu choice, or adds it to a variable and checks its content, as for a PIN or an account number. Because the variables are kept across events, the digits collected so far are available on each new event. The criteria to end the input (length, terminating key, timeout) are therefore defined in the service logic.

4.4 Timers

- **Set timeout** starts a timer, in seconds, for the call. When it expires, a timeout event is generated.
- **Cancel timeout** cancels the timer.

Timers are used to detect inactivity, for example to repeat a menu or release the call if no digit is entered.

4.5 Setting Up and Bridging Calls

- **Originate call** sets up a new call from the media server towards a destination in user@host form, with a configurable timeout. The calling number of the original call is used by default and can be replaced. Extra SIP headers can be added to the outgoing INVITE. The service logic is informed with an originated event when the call is answered, or an originate_failed event when the destination is busy or congested.
- **Bridge call** connects the caller with all the calls originated for it, in a single mixing bridge. Since SRE 4.2, more than two parties can be bridged (N-way bridging), which allows building a simple conference by originating calls to several participants.
- **Mute channel** and **Unmute channel** mute or unmute the audio of the caller, of the other parties or of both, in the incoming direction, the outgoing direction or both.

4.6 Interacting with Other Calls

Broadcast event sends a named event to a given call, identified by its channel identifier. The event is delivered to all media processors, and the one handling that call runs its service logic with this event. This lets one call act on another, for example to notify a waiting caller that an agent has become available.

4.7 Speech-to-Text

Speech to text records what the caller says and transcribes it. The recording stops after a maximum duration, after a period of silence or on a DTMF key, and can start with a beep. When the transcription is available, an stt_finished event is generated with the text, which the service logic can analyse or pass to a language model. This is described further in the SRE and AI feature description.

5 Typical Services

- **IVR**: a greeting, a menu driven by DTMF or by speech, and a transfer to the right destination, with menus and destinations that can come from the data model.
- **Announcements**: playing a fixed or generated message, such as “number not in service” or a personalised notification, and releasing the call with the appropriate cause.
- **Click-to-call**: a call is set up towards a first party, and once it answers, a second call is originated and the two are bridged.
- **Call screening**: the called party is called first and hears an announcement about the incoming call, then accepts it with a key, which bridges the call, or refuses it, which releases the caller with a chosen cause.
- **Conference and queuing**: several parties are bridged in one call, music on hold is played while waiting, and other calls are notified with broadcast events.

6 Example: Simple IVR

The following media service logic implements a small IVR. On a SIP Call Processor, a SIP service logic identifies the service and uses **Relay to media server** to send the call to a media server, which runs this media service logic.

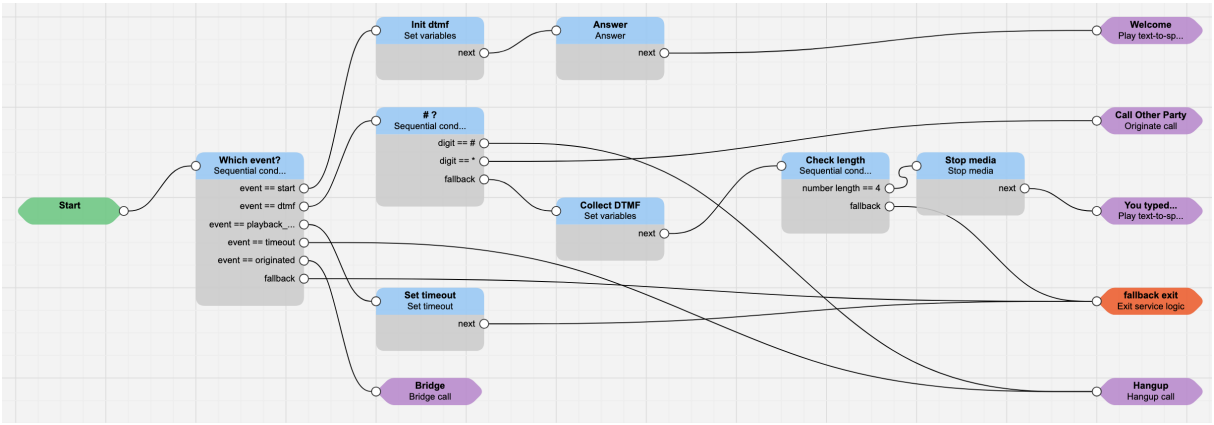


Figure 1: Sample media service logic

The service behaves as follows:

1. **start**: a variable holding the collected digits is reset, the call is answered and a text-to-speech greeting invites the caller to enter a 4-digit PIN, press * to reach an operator or # to end the call.
2. **playback_finished**: after each announcement, a timeout is set so that an inactive call does not stay open.
3. **dtmf**:
    - # releases the call.
    - * originates a call to the operator.
    - Any other digit is added to the collected digits. When four digits have been entered, the current announcement is stopped and a text-to-speech message reads the PIN back to the caller.
4. **originated**: when the operator answers, the caller and the operator are bridged.
5. **timeout**: the call is released.

Each event leads to a separate execution of the same service logic, and the collected digits are kept between them. In a real service, the PIN would be checked against the data model or an external system with the usual query nodes, and the operator's address would come from the data rather than being written in the service logic.