This document presents a reference design with a corresponding ser.cfg to enable you to quickly set up SER to do what most people would like SER to do.

Stateful vs. stateless

An often misunderstood concept is stateful vs. stateless processing. The description of the INVITE SIP transaction above is an example of stateful processing. This means that SER will know that the OK belongs to the initial INVITE and you will be able to handle the OK in an onreply_route[x] block. With stateless processing (or simply forwarding), each message in the dialogue is handled with no context. Stateless forwarding is used for simple processing of SIP messages like load distribution.

In order to implement any advanced functionality like call accounting, forward on busy, voicemail, and other functionality in this documents reference setup, you will need to use stateful processing. Each SIP transaction will be kept in SERs memory so that any replies, failures, or retransmissions can be recognized. One consequence is that when using t_relay() for a SIP message, SER will recognize if a new INVITE message is a resend and will act accordingly. If stateless processing is used, the resent INVITE will just be forwarded as if it was the first.

The confusion arises because this stateful processing is per SIP transaction, not per SIP dialog (or an actual phone call)! A phone call (SIP dialog) consists of several transactions, and SER does not keep information about transactions throughout a particular phone call. The consequence is that SER cannot terminate an on-going call (it doesnt know that the call is on-going), nor can SER calculate the length of an ended call (accounting). However, SER can store when an INVITE (or ACK) and a BYE message are received and record this info together with the Call-Id. A billing application can then match the INVITE with the BYE and calculate the length of the call.

Understanding SIP and RTP

In order to understand the subsequent sections, you must understand a few things about SIP and RTP. First of all, SIP is a signalling protocol handling the call control like inviting to a call, cancel (hang up while ringing), hanging up after ended call and so on. SIP messaging can be done directly from user agent to user agent, but often a SIP server is needed for one user agent to find another.

When a SIP server like SER receives a message, it can decide that it wants to stay in the loop or not. If not, SER will provide the user agents with the information they need to contact each other and then SIP messages will go directly between the two user agents.

If SER wants to stay in the loop, for example to make sure that a BYE message is received for accounting, SER must insert a Route header in the SIP message (using the record_route() function) to tell everybody that it wants to participate. In order for this to work, SER and potentially other SIP servers who participate must do what is called loose routing. A bit simplified, it means that SIP messages should not be sent directly to the user agent, but rather indirectly via all who have put a Route header in the SIP message (to check for such a recorded route, the loose_route() function is used).

SIP can also include additional information, for example related to how to set up a call with an audio or video stream (called SDP, Session Description Protocol). The SDP information will result in one or more RTP streams (or sessions) to be set up, normally directly between the two user agents. SER will NEVER participate in the RTP stream. RTP streams are by nature bandwidth and processing intensive. However, as we will describe later, SER can make sure that a third party application like a B2BUA or RTP Proxy can become the middle man.

Finally, the Real-Time Control Protocol (RTCP) communicates information about the RTP streams between the user agents (RTCP will either use the specified RTP port + 1 or the port indicated in the SDP message).

Back-end applications and B2BUA

We have learned that SER does not keep the state of a dialog, but just transfers SIP messages (as part of transactions) between two user agents. The consequence is that SER cannot be a peer in a dialog.