Programmer Guide: Dialog Usage Manager

Table of Contents

Concepts 1
Major Components 1
Definitions 1
Handling Invite Sessions - Client 2
Handling Invite Sessions - Server 2
Handling Registration - Client 2
Handling Registrations - Server 2
Handling Subscriptions - Client 2
Handling Subscritpions - Server 2
Handling Publictionas - client 2
Handling Publications - Server 2
Dealing with Refer 3
Dealing with complex offer/answer 3

Concepts

Major Components

The DialogUsamgeManger (or DUM) is the unit that keeps track of all the data sturcutres and sits on top of the transaction layer of the stack. It keeps track of multiple DialogSet which contains Dialogs. Each DialogSet contains all the Dialog that were created by a common initial request. They all share the same SIP Call-Id and from tag from the orginal request. Inside a specific dialogSet there can be some type of BaseCreator that represnets the intial request that generated the dialog. This will only exist onthe UAC side. The DialogSet also contains serveral objects dereived from BaseUsage that are using this particular dialog. There are several types of things that are a Usage of this dialog. There can be one InvSession, one Registration, one Publication, multiple Subscriptions and multiple OutOfDialogRequests. Note the name OutOfDialog is a little weird - they are actually in a thing a lot like a dialog but are transactions that are not in one of the other categories. Typically messages that result in OutOfDialogRequests are MESSAGE and OPTIONS.

An initial SIP Request is created by calling the makeX interfaces on the DUM. It is then sent using the send interface on the DUM. This will create some internal data structures and return a DialogSetID that can be used to find all the state asscociates resulting from this. When a response comes into this, a call-back from one of the Handler classes will be called to notify the application about incoming events. This will pass up some type of client or server usages class that can be used to send aditional messages and responses inthe context of this particularr usage of the Dialog.

Definitions

DialogUsage Manager - Main class that keeps track of all the DialogsSets, Dialogs, and Usages.

DialogSet - A set of dialogs that were gerneated from a common request. They share the same call-id and the same from tag in the request that genreated the dialog.

Dialog - A container holding such things and local and remote CSEQ, URI, Call-ID and such as defined by the SIP standard.

DialogID - An identifier that uniquely finds a a Dialog by Call-ID, and to and from tags.

DialogSetID - An identifier that uniquily identifies a Dialog-Set and is formed from Call-ID and

Usages - These are the objects are are using a dialog. They include ClientInviteSession, ClientOutOfDialogReq, ClientPublication, ClientRegistration, ClientSubscription, ServerInviteSession, ServerOutOfDialogReq, ServerPublication, ServerRegistration, and ServerSubscription. These have varios operations that can be called on them to

Handlers - These are objects uses to derive class from that allow callbacks from this layer to the application using ti. They include InviteSessionHandler, OutOfDialogHandler, RegistrationHandler, SubscribeHandler, and PublictionHandler.

Handles - All the Usaves and Handlers are not really exposes to the applications using this layer. Instead, handles to them are passed out. When the applications goes to use a handle, the unerlying object may have been delted and the applications must be prepared for this not to work.

Handling Invite Sessions - Client

Initially a cleint can call makeNewInvite on the DUM and get a SipMessage. It can take this and modify it such as adding SDP. It then calls sends on the DUM and sends themessage. This will cause the creation of a DialogSet for the Dialogs resulting from this request. When a responses (such as a 180) comes back that cuase the creation of an early dialog, the onEarly callback in the Handler willbe called. This will pass in a ClientInviteSession to the applications and the eactual message received. When and offer or answer is recived, the onAnswer or onOffer callback will be called. In the typical case wher the INVITE sent and offer and the 180 has SDP but it is not an answer, only the onEarly will be called. If the 180 was reliable so that it was an aswer, then both the onEarly and the onAnswer would be called.

If the client which to sent a new answer or offer it must call the setAnswer or setAnswer object with the new SDP. This saves it but does not send a messages. the client then calls sendAnyAnswer or sendAnyOffer to cause the aproperate message to be sent to send a new offer or answer. This might be a PRACK, an UPDATE, a reINVITE depending on the currenst state of the dialog.

Handling Invite Sessions - Server

Handling Registration - Client

Handling Registrations - Server

Handling Subscriptions - Client

Handling Subscritpions - Server

Handling Publictionas - client

Handling Publications - Server

Dealing with Refer

Dealing with complex offer/answer

When you receive an offer, you need to send an aswer. If you don't like something about the SIP message with the offer you can totally reject it at the sip level, but if you don't like the media it proposed, you need to send a answer with all the m lines you don't like zeroed out. You can then instantly send a counter offer to propose somethign new.