Programmer Guide: Dialog Usage Manager

Table of Contents

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, ClientOutOf-DialogReq, 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, Subscribe-Handler, 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

When the sserver first recives an INVITE, it will call the onNewInvSession handler and pass a handle to a ServerInviteSession. It the UA is busy this could be rejected with the reject mehtod. If the INVITE contaned an offer, the onOffer callback gets called. When this happens, the server needs to call the setAnser function to set the new offer. This will not send the offer. The server must then call the sendAnserInAnyMessage to actually send the offer. This sequence of two calls seem a little weired but is required to make the all the cases work when doing PRACK stuff.

From this pont on thir can be several rounds of sending and receivng offers and aswers. When the UA which to sendd a 200 and "answer" the call, it calls the accept method. Finially when it wishes to end the call it calls the end method. If the other side ends the call, the onTerminated callback will get called.

INFO messages can be received in the dialog with the onInfo callback. Refer is a topic all on it's own and covered later in this document.

Handling Registration - Client

The client froms a new registration message by calling makeResitration in the DUM. This returns a SIP

messages which is sent by calling send on the DUM. This will initate the registrtionprocess.

If it succeedes or failes, the aproperate onSuccess or onFailure will be called. The DUM will contintue to run the times and keep this registration alive. This may result in an onFailure callback at any time.

The callback passes back a handle to a ClientRegistration which provies serveral methods for the client to discover and manipulate the contacts for the registration. myContacts refers to contatst this UA has registered while allContacts referes to any contact that has been registered for this AOR. The add and remove binding calls allow for manipulation of contacts this UA registered and for cotacts that other UA registered.

Handling Registrations - Server

When a new registgration is received the onAdd will be called. The server needs to either accept this on rejct it using the coresponding function isn the ServerRegistration class. If the registration is refreshed byt the client onRefresh will be called and if it expereis before it is refreshed, onExpired will be called. If the client removes on of the the contacts, onRremoveOne will be called and if the client removes all the contacts, onRemoveAll be be called.

Handling Subscriptions - Client

Handling Subscritpions - Server

Gets a onNewSubecritpion callabck when it is created and onRefresh or OnTerminated as the clients updates it.

Handling Publictionas - client

Handling Publications - Server

Dealing with Refer

Dealing with Digest authentication

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.