Call Class

A wrapper object in the voice chat example that bundles groups of **Session** objects into a single chat connection. It provides wrapper methods to handle adding or removing sessions, muting, and monitoring. It also manages creating a new **VoiceChannel** object (which wraps a **P2PTransportChannel** object and a **MediaChannel**) and associating that with each session. This object is created by **PhoneSessionClient**. To destroy it, call **PhoneSessionClient::DestroyCall** (the **Call** object calls this automatically when a **Session** is removed and the **Session** count reaches zero). The methods shown are not thread-safe, and can be called on any thread.

Syntax

```
class Call : public MessageHandler,
             public sigslot::has_slots<>
```

Methods

The following public methods are exposed by **Call**.

Name	Description
void **AcceptSession**(Session **session*)	Accepts an incoming connection request. *session* is the **Session** sent out previously by **SignalSessionState**.
Call(PhoneSessionClient **session_client*)	Constructor. Creates a random ID to identify this object.
~Call	Destructor. This method destroys all the **Session** objects that it holds a pointer to.
uint32 **id**()	Returns a random ID assigned to the object (this is not the JID).
Session* **InitiateSession**(const buzz::Jid &*jid*, vector<XmlElement*> **extra_xml*)	Called by **CallClient::MakeCallTo** to explicitly make a call to someone. It sends a connection request and a PhoneSessionDescription (a list of codecs) to the other party, and starts generating local candidates. *extra_xml* is just an extra blob of arbitrary XML to send along with the stanza sent to the other computer (note that this is not used for session description or session info stanzas; this parameter is not used in any of the example code).
void **Mute**(bool *mute*)	Mutes the sound in all the **Session** objects.
bool **muted**()	Whether or not the sound is muted in the **Session** objects.
void **OnAudioMonitor**(VoiceChannel **channel*, const AudioInfo &*info*)	Called by the the **Session VoiceChannel** objects periodically with audio monitoring data. Sends **SignalAudioMonitor**. This is not called unless **StartAudioMonitor** is called.
void **OnMessage**(Message **message*)	Called with commands from other threads to enable multithreading. The only message handled is to call **SessionClient::DestroyCall** when the object no longer contains any **Session** objects.

void **RedirectSession**(Session **session*, const buzz::Jid &*to*)	Sends a redirect reply to a session connection request. This is a **Session** sent out previously by **SignalSessionState**.
void **RejectSession**(Session **session*)	Rejects a session connection request for the session sent out previously by **SignalSessionState**.
std::vector<Session *> & **sessions**()	Returns the list of **Session** objects managed by this call.
void **StartAudioMonitor**(Session **session*, int *cms*)	Causes **SignalAudioMonitor** signals to be sent. This is not called in libjingle as shipped.
void **StartConnectionMonitor**(Session **session*, int *cms*)	Causes **SignalConnectionMonitor** signals to be sent. This is not called in libjingle as shipped.
void **StopAudioMonitor**(Session **session*)	Ends the sending of **SignalAudioMonitor** signals.
void **StopConnectionMonitor**(Session **session*)	Ends the sending of **SignalConnectionMonitor** signals.
void **Terminate**()	Destroys all the sessions by calling **TerminateSession** on each **Session**.
void **TerminateSession**(Session **session*)	Calls **Session::Terminate** on each **Session**, which releases its resources.

Signals

SignalAddSession< Call *, Session * >
Called when **AddSession** is called and a session is added. No objects currently subscribe to this signal.

SignalRemoveSession< Call *, Session * >
Called when **RemoveSession** is called and a session is removed. No objects currently subscribe to this signal.

SignalSessionState< Call *, Session *, Session::State >
Called when a **Session** sends state information. It is very important to subscribe to this signal because it alerts the application when call connection requests are received, when calls are terminated, or other important events. Subscribe to this signal as soon as **PhoneSessionClient** sends **SignalCallCreate**.

SignalSessionError< Call *, Session *, Session::Error >
Called when a **Session** object sends an error message.

SignalConnectionMonitor< Call *, Session *, const std::vector< ConnectionInfo > & >
Sends periodic connection monitoring information. This signal won't be sent unless you call **StartConnectionMonitor**.

SignalAudioMonitor< Call *, Session *, const AudioInfo & >
Sends periodic audio monitoring information. This signal won't be sent unless you call **StartAudioMonitor**.

Friends

- PhoneSessionClient

Attributes: public

Declaration file: talk/session/phone/call.h

Last updated March 23, 2012.