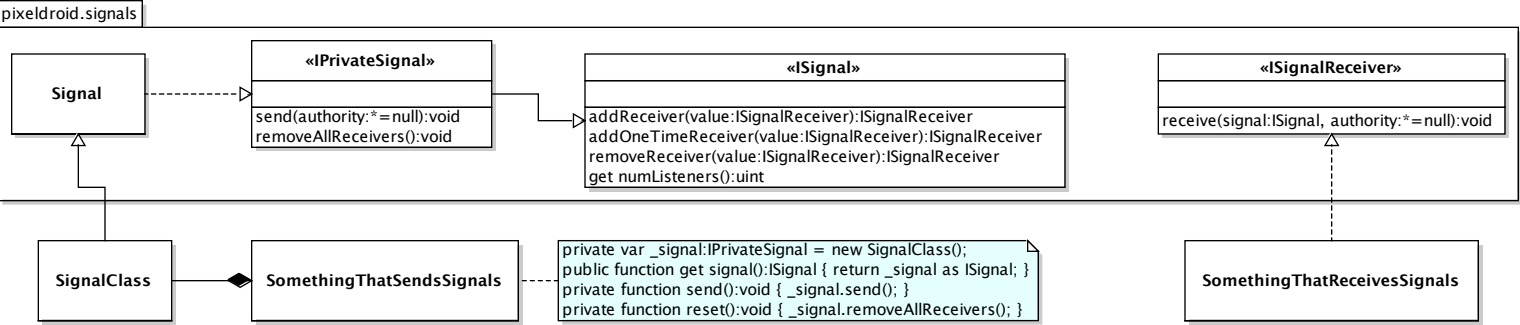Used directly, Signal and SignalReceiver provide a simple publication / subscription interface.
Classes can aggregate signals to declare the events they publish
PrivateSignal provides elevated access for signal owners
A Signal authority is generally whatever sent the signal, but may be anything useful, including nothing at all

pixeldroid.signals

Signal

«IPrivateSignal»

send(authority:*=null):void
removeAllReceivers():void

«ISignal»

addReceiver(value:ISignalReceiver):ISignalReceiver
addOneTimeReceiver(value:ISignalReceiver):ISignalReceiver
removeReceiver(value:ISignalReceiver):ISignalReceiver
get numListeners():uint

«ISignalReceiver»

receive(signal:ISignal, authority:*=null):void

SignalClass

SomethingThatSendsSignals

private var _signal:IPrivateSignal = new SignalClass();
public function get signal():ISignal { return _signal as ISignal; }
private function send():void { _signal.send(); }
private function reset():void { _signal.removeAllReceivers(); }

SomethingThatReceivesSignals

The SignalController provides a notifier / observer approach to delegation.
Senders broadcast signals to zero or more receivers.
A SignalReceiver may be thought of as a Command, with the receive() method functioning as execute()

pixeldroid.signals

«ISignalController»

addSender(sender:ISender, signal:ISignal):void
removeSender(sender:ISender signal:ISignal):void
addReceiver(receiver:ISignalReceiver, signalClass:ISignal):void
removeReceiver(receiver:ISignalReceiver, signalClass:ISignal):void
removeReceiversForSignal(signalClass:ISignal):void

SignalController

ViewClassA → addSender(this, uiSignal) → **ControllerClass**

ControllerClass ← addReceiver(this, MODEL_UPDATE) ← **ViewClassB**

addReceiver(this, UI_GESTURE)

addSender(this, modelSignal)

CommandClass

ModelClass

SignalFactory
public const UI_GESTURE_SIGNAL_CLASS
public const MODEL_UPDATE_SIGNAL_CLASS
public function get uiSignal():ISignal;
public function get modelSignal():ISignal;