WebRTC Error Reporting

Interim Meeting: May 2014

Eric Rescorla

Mozilla

ekr@rtfm.com

General Principles

- Error types
  - Programming errors get exceptions
  - Other errors get callbacks
- WebIDL does type checking
- Try not to fail silently

Calls after `pc.close()`

- Some methods generate `InvalidStateError` (e.g., `addStream()`) and some don't (e.g., `createAnswer()`)
- This is confusing for programmers
- Proposed behavior
  - Any call other than `close()` generates `InvalidStateError`
  - Asynchronous calls followed by `close()` simply never happen
    - Some of the descriptions have them happen but then not generate callbacks
  - `close()` fails silently
- This should all be written in one place

Multiple calls to `addStream()` with same stream

- Currently this is ignored
- Proposed behavor
  - Throw `ResourceInUse`

Trying to send on a closed DataChannel (§ 5.2.)

- Currently behavior
  - Increment `bufferedAmount` (step 3)
  - Then silently abort
- Proposed behavior
  - Throw `InvalidStateError`

Create RTCDTMFSender on non-audio track

- Unclear what current behavior is
  - Probably create it with `.canInsertDTMF === false`
- Proposed behavior
  - Generate an `InvalidParameter` exception

Call `insertDTMF()` when `.canInsertDTMF` is false

- Current behavior
  - Silently ignore
- Proposed behavior
  - Generate `InvalidStateError`

Insert DTMF with bogus values (Guduru)

- Generate `InvalidParameter` exception

Unused errors (§ 4.6.2.1)

- `IncompatibleConstraintsError`
- `incompatibleMediaStreamTrackError`
-
- Propose removing these

`RTCSdpError` (§ 4.6.2

- This is defined but never used
  - Even though it's semi-referenced
- Proposed resolution
  - Edit `set{Local,Remote}Description` sections to make clear it is passed to error callbacks

`addIceCandidate()` **in wrong state**

- Claim 1: this should be queued
- Claim 2: Any attempt to `addIceCandidate()` in wrong state should get `InvalidSteError` (just like everything else)

`addIceCandidate()` **bogus data**

"If the candidate could not be successfully applied, the user agent must queue a task to invoke failureCallback with a DOMError object whose name attribute has the value TBD (TODO InvalidCandidate and InvalidMidIndex)."

NOTE What errors do we need here? Should we reuse the *SessionDescriptionError names or invent new ones for candidates? Should this method be queued?

- The session description errors just say "it's busted" and give line number
  - Assume we don't want to change that
  - And even if they did, it would just say "ICE candidate is busted"
- Proposed resolution
  - `InvalidCandidate`, `InvalidMid` and `InvalidMLineIndex` with obvious meanings

failure callbacks for `createOffer()` and `createAnswer()`

If the SDP generation process failed for any reason, the user agent must queue a task to invoke failureCallback with an DOMError object of type TBD as its argument.

- Can anything go wrong here?
- ... other than `InternalError`?
- Can pretty much be called in any state

failure callbacks for `setLocalDescription()` and `setRemoteDescription()`

- Currently the following errors
  - `InvalidSessionDescriptionError` — SDP is bogus; need to specify line number
  - `IncompatibleSessionDescriptionError` — can't apply SDP; freeform text to explain the problem?
- Also add
  - `InvalidStateError` — e.g., `setLocalDescription(answer)`
  - `InvalidPeerIdentityError` — e.g., if the target identity doesn't match the remote description
  - What should we use for bogus changes to SDP? `InvalidSessionDescriptionError`?

Spontaneous Errors

- Some kinds of errors just happen
  - TURN errors
  - DTLS connection error
  - "Internal errors"
- These aren't tied to any API call

`.onerror` **or something**

- Throws some typed object

```
interface RTCRuntimeError : DOMError {
    readonly    attribute explanation;
};
```

- The `.name` property should tell us what happened (machine readable)
- The `.explanation` should be freeform
- Individual errors could have their own attributes
  - E.g., TURN server URL for TURN errors
  - Idea is that it should be processable by app

What about state?

- How do I distinguish fatal from nonfatal
- State needs to change *first*
- So that the event handler can interrogate it

Are there enough of these?

- TURN errors
- DTLS connection error
- “Internal errors”
-
- Are there really enough of these to design a general facility?