RTCDataChannel

This is an experimental technology
Because this technology's specification has not stabilized, check the compatibility table for the proper prefixes to use in various browsers. Also note that the syntax and behavior of an experimental technology is subject to change in future versions of browsers as the spec changes.

The **`RTCDataChannel`** interface represents a bi-directional data channel between two peers of a connection.

Objects of this type can be created using `RTCPeerConnection.createDataChannel()`, or are received in a `datachannel` event of type `RTCDataChannelEvent` on an existing `RTCPeerConnection`.

On Gecko, this API is called `DataChannel` instead of the standard `RTCDataChannel` name.

Properties

`RTCDataChannel.label` Read only
Returns a `DOMString` containing a name describing the data channel. There is no constraint of uniqueness about it.

`RTCDataChannel.ordered` Read only
Returns a `Boolean` indicating if the order of delivery of the messages is guaranteed or not.

`RTCDataChannel.protocol` Read only
Returns a `DOMString` containing the name of the subprotocol in use. If none, it returns "".

`RTCDataChannel.id` Read only
Returns an `unsigned short` being a unique id for the channel. It is set at the creation of the `RTCDataChannel` object.

`RTCDataChannel.readyState` Read only

Returns an enum of the type `RTCDataChannelState` representing the state of the underlying data connection. It can be one of the following values:

- `"connecting"` is the state indicating that the underlying connection is not yet set up and active. This is the initial state of a data channel created with `RTCPeerConnection.createDataChannel()`.
- `"open"` is the state indicating that the underlying connection is up and running. This is the initial state of a data channel dispatched in a `RTCDataChannelEvent`.
- `"closing"` is the state indicating that the underlying connection is in the process of shutting down. No new sending task is accepting but the cached messages are in the process of being sent, or received.
- `"closed"` is the state indicating that the underlying connection has been shut down (or couldn't be established).

`RTCDataChannel.bufferedAmount` Read only

Returns an `unsigned long` containing the amount of bytes that have been queued for sending: that is the amount of data requested to be transmitted via `RTCDataChannel.send()` that has not been sent yet. Note that if the channel is `closed`, the buffering continues.

`RTCDataChannel.bufferedAmountLowThreshold`

Is an `unsigned long` representing the number of bytes at which the `RTCDataChannel.bufferedAmount` is considered to be low. When the `RTCDataChannel.bufferedAmount` decreases from above this threshold to equal or below it, the `bufferedamountlow` event fires. This value is initially zero on each new channel object, and the application may change this value at any time.

`RTCDataChannel.binaryType`

Is a `DOMString` indicating the type of binary data transmitted by the connection. This should be either `"blob"` if `Blob` objects are being used or `"arraybuffer"` if `ArrayBuffer` objects are being used. Initially it is set to `"blob"`.

`RTCDataChannel.maxPacketLifeType` Read only

Is an `unsigned short` indicating the length in milliseconds of the window in when messaging happens in unreliable mode.

`RTCDataChannel.maxRetransmits` Read only

Is an `unsigned short` indicating the maximum amount of retransmissions that can happen when messaging happens in unreliable mode.

`RTCDataChannel.negotiated` Read only

Is a `Boolean` indicating if the channel has been negotiated by the application, or not.

DataChannel.reliable ⚠ Read only

Is a `Boolean` indicating if the connection can send message in unreliable mode.

DataChannel.stream ⚠ Read only

Is an obsolete synonym for `RTCDataChannel.id`.

Event Handlers

RTCDataChannel.onopen

Is the event handler called when the `open` event is received. Such an event is sent when a the underlying data transport, that is the data connection, has been established.

RTCDataChannel.onmessage

Is the event handler called when the `message` event is received. Such an event is sent when a message is available on the data connection.

RTCDataChannel.onbufferedamountlow

Is the event handler called when the `bufferedamountlow` event is received. Such an event is sent when `RTCDataChannel.bufferedAmount` drops to less than or equal to the amount specified by the `RTCDataChannel.bufferedAmountLowThreshold` property.

RTCDataChannel.onclose

Is the event handler called when the `close` event is received. Such an event is sent when the underlying data transport has been closed.

RTCDataChannel.onerror

Is the event handler called when the `error` event is received. Such an event is sent when an error has been encountered.

Methods

RTCDataChannel.close()

Closes the channel. The closing is done in a non abrupt way. The `state` of the channel is set to `"closing"`, the messages not yet sent are sent, then the channel is closed.

RTCDataChannel.send()

Sends the data in parameter over the channel. The data can be a `DOMString`, a `Blob`, an `ArrayBuffer` or an `ArrayBufferView`.

Example

```
var pc = new RTCPeerConnection();
var dc = pc.createDataChannel("my channel");

dc.onmessage = function (event) {
  console.log("received: " + event.data);
};

dc.onopen = function () {
  console.log("datachannel open");
};

dc.onclose = function () {
  console.log("datachannel close");
};
```

Specifications

Specification	Status	Comment
WebRTC 1.0: Real-time Communication Between Browser The definition of 'RTCDataChannel' in that specification.	WD Working Draft	Initial specification.

Browser compatibility

Desktop | Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	(Yes)	(Yes) [1]	Not supported	(Yes)	?
`onbufferedamountlow`	Not supported	Not supported [1]	Not supported	33	Not supported

[1] The interface is called `DataChannel` and not `RTCDataChannel`

See also

- WebRTC