**Client**

**Log**

**Api**

**TCP Manager**

**UDP Manager**

**Network**

**Radio**

## Dynamic Models

## Static Models

**UDP Stream (Radio —> Client)**

| | | |
|---|---|---|
| **Panadapter** | **Waterfall** | **IqStream** |
| **AudioStream** | **Meter** | **MicAudioStream** |

**UDP Stream (Client —> Radio)**

**TxAudioStream**

**UDP Stream (Client <—> Radio)**

**Opus**

**Without UDP Stream**

| | | |
|---|---|---|
| **Slice** | **Amplifier** | **Tnf** |
| **Equalizer** | **UsbCable** | |
| **Memory** | **Xvtr** | |

| | |
|---|---|
| **Atu** | **Cwx** |
| **Gps** | **Interlock** |
| **Profile** | **Transmit** |
| **Wan** | **Waveform** |

# Dynamic Model Objects

## With UDP stream

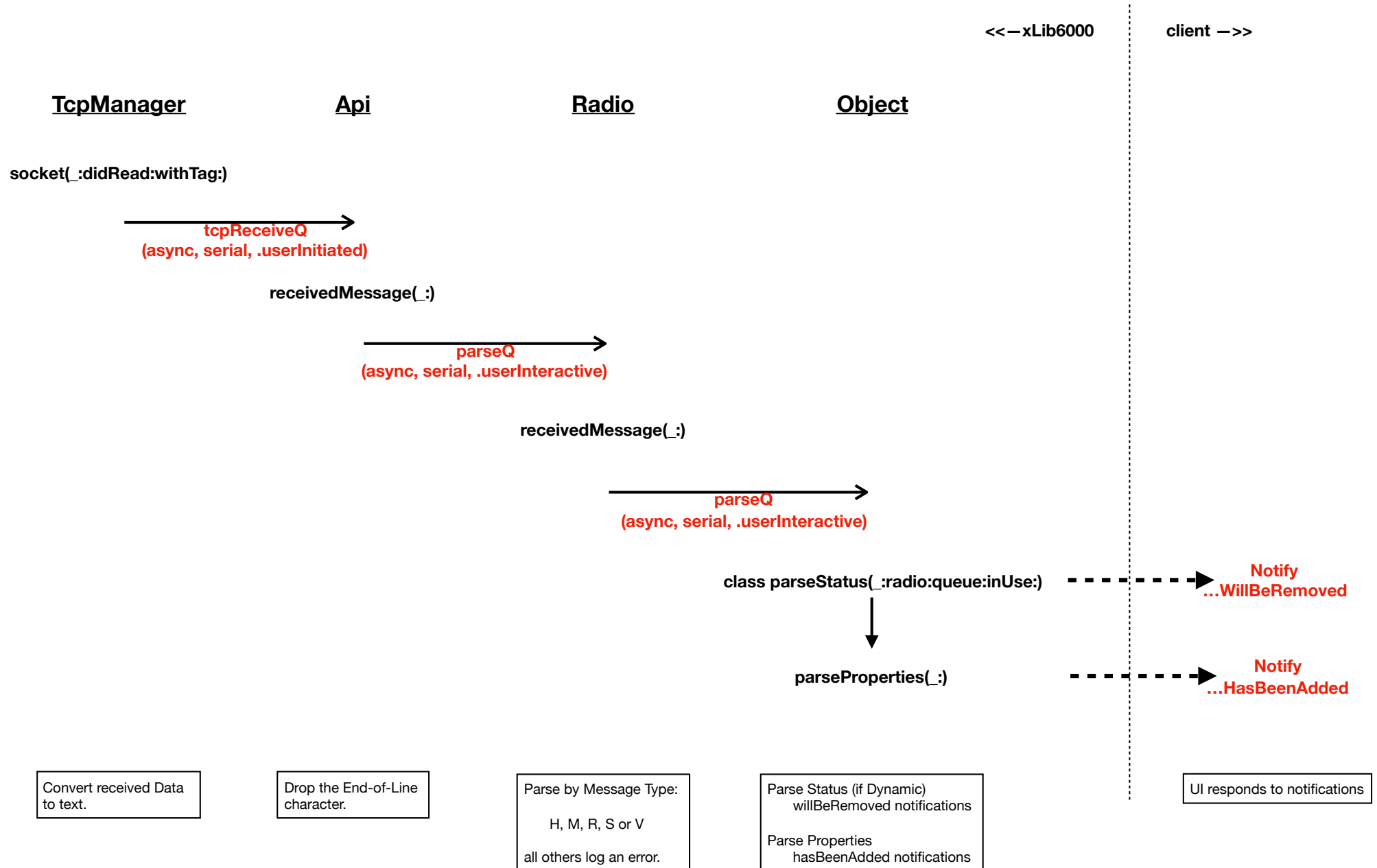| **Init( id: queue:)** |
|---|
| Public properties: <br><br>     private(set) var id: |
| Custom methods: <br><br>     optional . . . |
| StatusParser protocol methods: <br><br>     class func parseStatus(_ keyValues: radio: queue: inUse:) |
| PropertiesParser protocol methods: <br><br>     func parseProperties(_ properties:) |
| VitaProcessor protocol methods: <br><br>     func vitaProcessor(_ vita:) |

## Without UDP stream

| **Init( id: queue:)** |
|---|
| Public properties: <br><br>     private(set) var id: |
| Custom methods: <br><br>     optional . . . |
| StatusParser protocol methods: <br><br>     class func parseStatus(_ keyValues: radio: queue: inUse:) |
| PropertiesParser protocol methods: <br><br>     func parseProperties(_ properties:) |

# Static Model Objects

| **Init( queue:)** |
|---|
| Public properties: <br><br>     optional . . . <br><br><br> |
| Custom methods: <br><br>     optional . . . <br><br><br> |
| PropertiesParser protocol methods: <br><br>     func parseProperties(_ properties:) <br><br><br> |

# TCP Stream path

**TcpManager**          **Api**          **Radio**          **Object**

**socket(_:didRead:withTag:)**

→ **tcpReceiveQ**
**(async, serial, .userInitiated)**

**receivedMessage(_:)**

→ **parseQ**
**(async, serial, .userInteractive)**

**receivedMessage(_:)**

→ **parseQ**
**(async, serial, .userInteractive)**

**class parseStatus(_:radio:queue:inUse:)** ┈┈┈▶ **Notify**
**…WillBeRemoved**

↓

**parseProperties(_:)** ┈┈┈▶ **Notify**
**…HasBeenAdded**

| Convert received Data to text. | Drop the End-of-Line character. | Parse by Message Type:<br><br>H, M, R, S or V<br><br>all others log an error. | Parse Status (if Dynamic)<br>    willBeRemoved notifications<br><br>Parse Properties<br>    hasBeenAdded notifications | UI responds to notifications |

# UDP Stream path

**<<—xLib6000**   **client —>>**

**UdpManager**   **Api**   **Radio**   **Dynamic Model w/UDP**   **Client component**

**udpSocket(_:didReceive: ...)**

→ **udpReceiveQ**
**(async, serial, .userInitiated)**

**udpStreamHandler(_:)**

→ **udpReceiveQ**
**(async, serial, .userInitiated)**

**vitaParser(_:)**

→ **streamQ**
**(async, serial, .userInteractive)**

**vitaProcessor(_:)**

→ **streamQ**
**(async, serial, .userInteractive)**

**streamHandler(_:)**

| Parse bt Vita PacketType: | No processing: | Parse by Vita ClassCode, | Convert to <Object>.frame, | Use the Stream data |
| IfDataWithStream | | Validate by StreamId, | Check for missed packets, | |
| ExtDataWithStream | | Pass to <Object>.vitaParser | Pass to <Object> UI | (e.g. Draw a Panadapter, |
| all others log an error. | | | | use an Audio / IQ stream, etc.) |