```
1       void read_handler(const boost::system::error_code& ec, std::size_t bytes_read)
2       {
3           size_t offset = 0;
4           beast::flat_buffer::const_buffers_type bufs = buffer_.data();
5           //net::const_buffer bufs = buffer_.data();
6
7           std::string data(boost::asio::buffers_begin(bufs), boost::asio::buffers_end(bufs
            ));
8
9           const __int16* sp2 = boost::asio::buffer_cast<const __int16*>(buffer_.data());
10  //        const __int16* sp2 = boost::asio::buffer_cast<const __int16*>(bufs);
11          if (!ec)
12          {
13              int nSamples = 0;
14              int nBins = 0;
15              int scale = 1;
16              int bytes_used = 0;
17              while (bytes_used < bytes_read)
18              {
19                  int spectrumBytes = nBins * sizeof(__int16);
20                  if (nSamples > 0)
21                  {
22                      for (int ipp = 0; ipp < nSamples; ipp++)
23                      {
24                          if (offset + spectrumBytes > bytes_read)
25                          {
26                              std::cout << "not enough samples" << std::endl;
27                              std::cout << offset << " " << bytes_read << " " <<
                                spectrumBytes <<std::endl;
28                              nSamples = 0;
29                              break;
30
31                          }
32                          std::cout << " * " << ipp << " ";
33
34                          for (int i = 0; i < nBins; i++)
35                          {
36                              std::cout << (sp2[offset + i] / float(scale)) << " ";
37                          }
38                          std::cout << "\n";
39
40                          bytes_used += spectrumBytes;
41                          offset += spectrumBytes;
42                      }
43                  }
44                  // read header and find out # of samples
45                  else
46                  {
47
48                      std::size_t split = data.find_first_of('\x1e', offset);
49                      if (split != -1)
50                      {
51                          std::cout << " Split / offset " << split <<" / " << offset << std
                            ::endl;
52                          std::string chk = data.substr(offset, split - offset);
53                          //std::cout << "++++++++++++++++++++++++++" <<
                            int((data.substr(split, 1)).c_str()) << std::endl;
54                          offset = split + 1;
55                          std::size_t start = data.find_first_of('\{');
56
57
58                          std::string chk2 = chk.substr(start, chk.size() - start);
59                          std::cout << "****" << chk2 << std::endl;
60
61                          boost::property_tree::ptree root;
62                          std::istringstream iss(chk2);
63                          boost::property_tree::read_json(iss, root);
64
65                          nSamples = stoi(root.get_child("samples").get_value<std::string
```

```
                        >());
66                      nBins = stoi(root.get_child("sampleSize").get_value<std::string
                        >());
67                      scale = stoi(root.get_child("scale").get_value<std::string>());
68
69                      std::cout << "# of samples: " << nSamples << std::endl;
70                      std::cout << "# of bins         : " << nBins << std::endl;
71                      std::cout << "# of bytes reads  : " << bytes_read << std::endl;
72                      std::cout << "  offset           : " << offset << std::endl;
73                      std::cout << "  scale            : " << scale << std::endl;
74
75                  }
76              }
77
78          }
79
80
81
82          std::cout << "--------------------------------" << std::endl;
83          buffer_.consume(bytes_read);
84
85          //do_read();
86      }
87      //else
88      //{
89      //    if (!_closing)
90      //    {
91      //        throw fatal_exception("Error on connection: " + ec.message());
92      //    }
93      //}
94  }
95
```