

## NEUROTOR-xx42

### Руководство по Эксплуатации

## Введение

Взаимодействие с Программой NEUROTOR-xx42 для программ сторонних разработчиков выполняется по сети через соединения TCP на порт по умолчанию 10230. Разработаны протоколы прикладного уровня IPCGLUE и P2, приводится в настоящем документе.

Внутренние настройки для работы NEUROTOR-xx42 задаются изменением файлов конфигурации iptel.xml и databoost.xml.



## 1. Размещение компонентов NEUROTOR-xx42

Основные компоненты ПО — каталог /usr/local/opt/iptel. Подкаталоги содержат компоненты:

- bin — загрузочные модули Программы;
- etc — файлы конфигурации Программы;
- lib — динамические библиотеки, часть из которых является функциональными модулями.

## 2. Работа со сторонними программами

Сторона программы NEUROTOR-xx42 считается далее сервером, другая сторона считается клиентом.

### 2.1. Внешний протокол прикладного уровня IPCGLUE

Протокол является двоичным. Описание структур данных протокола приводится в Приложении. Сообщения протокола имеют структуру `ipc_msg`. Поля сообщений структуры `ipc_msg` имеют кодировку little-endian. Поле `ipc_msg.size` указывает размер структуры, следующей в поле `ipc_msg.data`. Если `ipc_msg.size` равно 0, сообщение считается пустым.

Программа ожидает от клиентов установки последовательно двух соединений: для событий, и для данных.

Установка соединений для событий и для данных требует указания `ipc_msg.type = IPC_CONNECT` в первом сообщении каждого соединения.

Для соединений для событий в поле `ipc_msg.data` указывается строка "call\_control", поле `ipc_msg.size` равно 12.

Для соединений для данных в поле `ipc_msg.data` указывается строка "data", поле `ipc_msg.size` равно 4.

Поле `ipc_msg.src_id` задаётся сторонней программой при первом сообщении (IPC\_CONNECT) случайным образом и в дальнейшем должно сохраняться.

Поле `ipc_msg.dst_id` задаётся клиентом в значение IPCGLUE\_ID.

Поле `ipc_msg.src_id` программой NEUROTOR-xx42 устанавливается в значение IPCGLUE\_ID.

`ipc_msg.dst_id` программой NEUROTOR-xx42 устанавливается в то же значение, которое указала сторонняя программа в первом сообщении (IPC\_CONNECT).

Ответом на сообщения IPC\_CONNECT программа NEUROTOR-xx42 должна отвечать пустым сообщением IPC\_ACCEPT, означающим готовность NEUROTOR-xx42 к взаимодействию.

Если любое из двух установленных соединений, для событий или данных, разорвано любой из сторон на уровне транспорта TCP, второе соединение также должно быть разорвано.

Если любая из сторон прислала по установленному соединению сообщение IPC\_DISCONNECT, принявшая сторона обязана разорвать соединение на уровне транспорта TCP.

### 2.2. Внутренний протокол прикладного уровня P2

Протокол является двоичным. Описание структур данных протокола приводится в Приложении. Сообщения внутреннего протокола имеют формат структуры `p2_msg`. При этом в поле внешнего протокола IPCGLUE должно указываться `ipc_msg.type = IPC_DATA`. Поля сообщений структуры `p2_msg` имеют кодировку big-endian.

В поле `ipc_msg.size` указывается длина данных поля `p2_msg` плюс последующие за ним данные, при их наличии.

Нумерация каналов осуществляется следующим образом. На каждую телекоммуникационную сессию отводится 2 канала, чётный — условное направление от А к Б, и нечётный — условное направление от Б к А.

Чётный канал имеет вычисляется по формуле  $2 * (N - 1)$

Нечётный номер канала — по формуле  $2 * (N - 1) + 1$

, где N — номер наблюдаемой телекоммуникационной сессии в диапазоне от 1 до предельного числа одновременно наблюдаемых сессий по лицензии.

#### 2.2.1. Сообщения для событий клиент - сервер

В поле `p2_msg.opcode`, клиент указывает команды из числа перечисления `p2_msg_event_opcodes`. Дополнительно, в зависимости от значения поля `p2_msg.opcode`, в полях `p2_msg` указывается (Таблица 1):

Таблица 1

Содержимое <code>p2_msg.opcode</code>	Указать дополнительно	Пояснение
<code>P2_CC_TARGET_SET</code>	<code>p2_msg.id = ID цели</code>	После <code>p2_msg</code> — целевой номер, заканчивающийся нуль-символом. Поле <code>ipc_msg.len</code> контролирует конец целевого номера
<code>P2_CC_TARGET_DEL</code>	<code>p2_msg.flow = 0xffffffff;</code>	Удаление всех целей

### 2.2.2. Сообщения для событий сервер - клиент

Сервер в поле `p2_msg.opcode` указывает числа перечисления `p2_msg_event_opcodes`. В поле `p2_msg.flow` — указывается номер канала, в котором обнаружено событие, в поле `p2_msg.session` указывается идентификатор вызова, для которого произошло событие. Дополнительно, в зависимости от значения поля `p2_msg.opcode`, в полях `p2_msg` указывается (Таблица 2):

Таблица 2

Содержимое <code>p2_msg.opcode</code>	Указано дополнительно	Пояснение
<code>P2_CC_EVENT_CALLER_ID</code>	<code>p2_msg.info.type=P2_INFO_SINGLE</code> <code>p2_msg.info.len=&lt;длина строки&gt;</code>	После <code>p2_msg</code> строка в форме <code>&lt;атрибут&gt; &lt;значение&gt;</code>
<code>P2_CC_EVENT_HOOK_STATE</code>	<code>p2_msg.hs.hook</code> из числа <code>p2_cc_hook_state</code> <code>p2_msg.hs.direction = {0   1}</code> <code>p2_msg.hs.len=1</code> <code>p2_msg.hs.type</code> из числа <code>p2_cc_flow_type</code> <code>p2_msg.param2=ID сессии</code> После <code>p2_msg</code> - <code>p2_hook_descr hd</code> , где <code>hd.flow</code> =номер канала <code>hd.task</code> =номер цели <code>hd.type=p2_msg.hs.type</code>	

### 2.2.3. Сообщения для данных сервер - клиент

Сервер посылает `p2_msg.opcode` равным `P2_DATA_FLOW`. В поле `p2_msg.compression` устанавливается значения согласно Таблице 3.

Поля `p2_msg.timestamp` имеют смысл соответствующих полей структуры `FILE_TIME` Win32 API. Каждое очередное сообщение должно иметь поле `p2_msg.timestamp` увеличенным на значение времени, соответствующее длительности звуковых данных из предыдущего сообщения. Звуковые данные располагаются сразу за полем `p2_msg`, количество ограничивается полем `ipc_msg.size`.

Таблица 3

Содержимое <code>p2_msg.compression</code>	Значение
0	U-law, 8 кГц дискретизации
8	A-law 8 кГц дискретизации
25	ADPCM56 little-endian

### 2.2.4. Сообщения для данных клиент - сервер

Передача не предусматривается.

### 3. Файл конфигурации Программы iptel.xml

Файл конфигурации имеет формат XML описывает, какие модули используются в работе, и их настройки. Расположен в каталоге /usr/local/opt/iptel/etc.

Общие параметры задаются в узле <iptel>. Параметры и их значения указаны в Таблице 4. У используемых модулей есть общие параметры, которые указаны в Таблице 5.

Таблица 4

Узел	Описание
<log_level>	Уровень логирования, шестнадцатеричное число, являющееся битовой маской, значения битов приведены в Таблице 6.
<log_path>	Путь к месту расположения файлов с журналами работы Программы. Каждый файл имеет структуру названия: <модуль>.log.tmp.
<fastcgi_pass>	Необязательный параметр, описывает привязку FastCGI интерфейса для диагностики состояния Программы при технической поддержке.
<sniffer>	Узел, описывающий модуль ввода. См. в Таблицу 7.
<sip_dec>	Узел, описывающий модуль декодера SIP. См. Таблицу 8.
<attribute_manager>	Узел, описывающий модуль диспетчера атрибутов. См. Таблицу 9.
<ipc_server>	Узел, описывающий модуль взаимодействия с клиентом. См. Таблицу 10.

Таблица 5. Общие параметры всех модулей

Узел	Описание
<log_level>	Уровень логирования, шестнадцатеричное число, являющееся битовой маской, значения битов приведены в Таблице 6.
<cpu_affinity>	Необязательный параметр. Перечисляет номера процессоров, к которым должны привязываться потоки выполнения каждого их модулей. Допускается использование символов запятой для указания номеров отдельных процессоров, и дефиса для указания диапазонов номеров процессоров.
<numa_nodes>	Необязательный параметр. Перечисляет номера узлов NUMA, память которых должна выделяться при работе модуля, если компьютера имеет несколько процессорных сокетов.

Таблица 6. Значения битовых полей узлов параметра log\_level

Номер бита	Описание
0	Выводить в файл журнала сообщения об ошибках
1	Выводить в файл журнала сообщения о предупреждениях
2	Выводить в файл журнала сообщения с информацией о состоянии значимых объектов и переменных
3	Выводить в файл журнала сообщения о внешних событиях
4	Выводить в файл журнала сообщения с информацией о состоянии малозначимых объектов и переменных

Таблица 7. Значения параметров модуля ввода

Узел	Описание
<threads>	Количество рабочих потоков для параллельной работы при вводе данных из интерфейсов ethernet. При вводе из модуля ускорения обработки databoost или при чтении данных из файлов параметр игнорируется, а количество образуемых рабочих потоков вычисляется, как описано далее.
<eth_if>	Название интерфейса ethernet для ввода данных. Параметр может быть указан несколько раз для параллельного одновременного ввода данных из нескольких интерфейсов. Если параметр <threads> указан в значении более 1, вводимые данные будут разделяться между рабочими потоками модуля средствами ядра ОС
<databoost>	Описывает параметры сетевого соединения с модулем ускорения обработки databoost. Параметр может быть указан несколько раз для параллельного одновременного ввода данных из одного и того же внешнего модуля databoost. Параметр устанавливает IP-адрес и TCP-порт подключаемого модуля databoost, разделённые символом «:». IP-адреса должны повторяться, а номер TCP-портов должны различаться. На дополнительных узлах следует указать параметр silent=«. Каждый дополнительный узел предусматривает образование дополнительного рабочего потока исполнения.
<play_file>	Название файла для ввода данных. Файл должен быть формата .pcap. Параметр может быть указан несколько раз для параллельного одновременного ввода данных из нескольких файлов.
<play_cnt>	Количество повторов ввода файлов, указанных в параметрах <play_file>.
<data_streams_balancing>	В случае работы модуля в единственном рабочем потоке, задаёт, должен ли модуль отправлять пакеты сигнализации SIP с соблюдением потоков данных. Возможные значения 0 (нет) и 1 (да). Позволяет гарантировать соблюдение последовательности пакетов между любыми двумя сетевыми узлами при обработке декодером SIP, если он работает в несколько рабочих потоков. Если модуль ввода многопоточный, значение узла игнорируется.

Таблица 8. Значения параметров модуля декодера SIP

Узел	Значение
<port>	Номер порта TCP, UDP, SCTP для вводимых пакетов сигнализации. Параметр может повторяться неоднократно для задания нескольких портов.
<process_sctp>	Указывает, требуется ли обрабатывать трафик транспортного уровня SCTP с портами, перечисленными в узлах <port>. Возможные значения 0 (нет) и 1 (да).
<process_tcp>	Указывает, требуется ли обрабатывать трафик транспортного уровня TCP с портами, перечисленными в узлах <port>. Возможные значения 0 (нет) и 1 (да).
<process_udp>	Указывает, требуется ли обрабатывать трафик транспортного уровня

Узел	Значение
	UDP с портами, перечисленными в узлах <port>. Возможные значения 0 (нет) и 1 (да).
<max_call_timeout_sec>	Максимально допустимая длительность телекоммуникационной сессии, в секундах. При превышении длительности сессии, сессия считается законченной, занятые ей ресурсы освобождаются. Если указать значение 0, длительность сессии не ограничивается.

Таблица 9. Значения параметров модуля менеджера атрибутов

Узел	Значение
<license_file>	Указывает путь доступа к файлу лицензии
<license_sign>	Указывает путь доступа к файлу подписи лицензии

Таблица 10. Значения параметров модуля взаимодействия с клиентом

Узел	Значение
<listen>	Указывает сетевой адрес и номер TCP-порта для ожидания подключений клиента.
<max_ts_lead>	Указывает максимально допустимый скачок отметок времени между RTP-пакетами, в 100-наносекундных интервалах, при превышении которого производится повторная синхронизация звуковых данных.

#### 4. Файл конфигурации Программы databoost.xml

Настоящий раздел приводится в ознакомительных целях, так как настройка описываемой части ПО NEUROTOR-xx42 сопряжено с достаточно кропотливой работой по учёту всех аппаратных возможностей оборудования Экспертизы.

Если интенсивность поступления пакетов непосредственно из сетевого интерфейса в ввода данных превышает несколько сотен тысяч единиц в секунду, модуль ввода данных может не успевает обрабатывать поступающие данные. В этом случае рекомендуется использовать входящее в состав Программы ускоритель ввода данных databoost.

С точки зрения операционной системы Databoost является дополнительным приложением, к которому модуль ввода данных подключается через TCP соединение. Databoost поддерживает сетевые контроллеры:

- Intel 82599;
- Intel x710;
- Intel x722;
- Intel e810;
- Intel 82599;
- Mellanox ConnectX-6DX.

Databoost обрабатывает пакеты на уровне пользователя без использования операционной системы. Если задано в файле конфигурации, и если поддерживает используемый сетевой контроллер, databoost может использовать аппаратные фильтры Flow Director, которыми сетевые контроллеры оснащены, для многократного уменьшения нагрузки на ядра процессоров.

Модуль ввода данных (Таблица 7) необходимо настроить так, чтобы не было задано ни одного параметра `<eth_if>` и `<play_file>`. Должен быть задан параметр `<databoost>`.

Общий формат командной строки databoost:

```
databoost <параметры EAL> -- -c <config_file_path>
```

Где `<config_file_path>` - путь доступа к файлу конфигурации.

`<параметры EAL>` - специфические параметры, необходимые для настройки работы среды EAL библиотеки DPDK, например, перечисление адресов PCI интерфейсов для используемых сетевых контроллеров. Возможный пример командной строки:

```
databoost --huge-dir=/hugetlbfs -l 10-11 -n 12 \
--file-prefix=rte_10000 \
-a 0000:41:00.0 -a 0000:41:00.1 -- \
-c /usr/local/opt/iptel/etc/databoost.json
```

В данном примере задано:

- использование памяти, отображённой на файловую систему hugetlbfs, находящейся в директории /hugetlbfs, для минимизации кэш-промахов при обработке пакетов;
- использование логических процессоров 10 и 11 для обработки полученных пакетов;
- подсказка библиотеке DPDK, что память системы является 12-канальной;
- уникальный префикс `rte_1000` имён файлов для файловой системы hugetlbfs — для избежания конфликтов имён с возможными другими приложениями библиотеки DPDK;
- использование устройства с адресами PCI 0000:41:00.0 и 0000:41:00.1 для ввода сетевых пакетов;
- использование файла /usr/local/opt/iptel/etc/databoost.json для настроек прочих параметров работы.

По результатам обработки командной строки и в дополнительных параметров в формате JSON, перечисленных в Таблице 11, Databoost выполняет действия:

- создаёт один или несколько потоков обработки (путём сопоставления с правилами фильтрации) пакетов из сетевых интерфейсов, причём каждый поток может обрабатывать пакеты из одного или нескольких интерфейсов;

- создаёт одно или несколько сетевых соединений и такое же количество потоков управления для приёма команд управления пакетными фильтрами и передачи отфильтрованных пакетов модулю ввода данных, причём для каждого сетевого соединения создаётся свой буфер FIFO для помещения в него отфильтрованных пакетов потоками обработки для передачи в модуль ввода данных;
- создаёт поток управления FastCGI-интерфейса для диагностики работы службой технической поддержки.

Таблица 11. Значения параметров в файле databoost.json

Параметр	Значение
«ports»	Массив элементов JSON, каждый из которых описывает настройки портов сетевых интерфейсов. В каждом элементе есть поля «id», «flow_director», и «packet_processor». Общее число элементов должно быть в пределах числа элементов -a <pci_address> командной строки.
«id»	Присваивает сетевому интерфейсу номер, целое число. Номер 0 соответствует самому первому интерфейсу, заданному параметром -a <pci_address> командной строки, номер 1 — следующему из них, и т. д.
«packet_processor»	Строковое значение, определяет алгоритм фильтрации пакетов после чтения из очереди сетевого контроллера. Возможные значения: <ul style="list-style-type: none"> <li>• «generic» - каждый пакет обрабатывается отдельно, независимо от других;</li> <li>• «simd64» - обрабатываются одновременно по 4 пакета, для архитектуры x86_64 — операциями блока MMX;</li> <li>• «simd128» - обрабатываются одновременно по 8 пакетов, для архитектуры x86_64 — операциями блока SSE2;</li> <li>• «simd256» - обрабатываются одновременно по 16 пакетов, для архитектуры x86_64 — операциями блока AVX;</li> <li>• «simd512» - обрабатываются одновременно по 32 пакета, для архитектуры x86_64 — операциями блока AVX512F.</li> </ul>
«flow_director»	Необязательный параметр, строковое значение. Указывает необходимость дополнительного использования средств поддержки аппаратной фильтрации сетевого контроллера Flow Director. Возможные значения: <ul style="list-style-type: none"> <li>• «e810» - для работы с контроллером Intel e810;</li> <li>• «x710» - для работы с контроллером Intel x710;</li> <li>• «i82599» - для работы с контроллером Intel 82599;</li> <li>• «mlx5» - для работы с контроллерами Mellanox семейства mlx5, например, ConnectX-6DX.</li> </ul> Если параметр не указан, используется фильтрация только средствами центрального процессора.
«listeners»	Массив элементов JSON, каждый из которых описывает ответную часть для связи с модулем ввода данных (см. описание параметра <databoost> в Таблице 7). На каждый описанный элемент создаётся отдельный поток, который выполняет: <ul style="list-style-type: none"> <li>• приём команд из модуля ввода на модификацию таблиц быстрой фильтрации пакетов;</li> <li>• отправку по сетевому соединению в модуль ввода данных</li> </ul>

Параметр	Значение
	пакетов, отобранных потоками обработки после получения из сетевых интерфейсов.
«bind»	В элементе описания ответной части для связи с модулем ввода данных, описывает прослушивающее сетевое соединение. Формат <адрес>:<port>, транспортный протокол TCP.
«affinity»	Целое число, означающее номер логического процессора для привязки работы потока работы ответной части связи с модулем ввода данных.
«fcgi_pass»	Определяет FastCGI интерфейс доступа службы технической поддержки для диагностики при жалобах пользователей. Формат <адрес>:<port>.
«fcgi_affinity»	Необязательный параметр, определяет привязку потока управления FastCGI интерфейса доступа службы технической поддержки. Возможные значения: <ul style="list-style-type: none"> <li>целое число — номер выделяемого логического процессора;</li> <li>строковое значение в кавычках - допускается использование символов запятой для указания номеров отдельных процессоров, и дефиса для указания диапазонов номеров процессоров.</li> </ul> Значение по умолчанию «0-1023».
«log_file»	Задаёт директорию для расположения файла журнала работы процесса Databoost. Значения по умолчанию нет, файл журнала не создаётся.
«fifo_descriptors»	Задаёт число элементов FIFO-буфера, в который потоки-обработчики принятых из сетевых контроллеров пакетов помещают пакеты, отобранные для передачи по сети модулю ввода.

Во время работы Databoost получает команды управления пакетными фильтрами от модуля ввода данных. На интерфейсах, на которых используется аппаратная поддержка фильтрации Flow Director, осуществляется выдача команд управления Flow Director путём записи в регистры управления соответствующим сетевым контроллером.

```

#ifndef __IPCGLUE_H
#define __IPCGLUE_H
#ifndef _WIN32
    #include <endian.h>
#else
    #define htobe32(x) (x)
    #define htobe16(x) (x)
    #include <order.h>
#endif
#include <sys/time.h>

#include <string.h>
#include <stdint.h>

#define TARGET_DEL      0
#define TARGET_ADD      1
#define TARGET_DEL_ALL  2
#define TARGET_EMPTY    0xffffffff
#define CH_ADD          0
#define CH_UPDATE       1
#define CH_REMOVE       2
#define CH_ADD_INFO     3
#define CH_UPDATE_INFO  4
#define CH_REMOVE_ONE   5

#define IPCGLUE_ID      24851

#define P2_MSG_VERSION  0
#define P2_EXT_FLOW_NOEMAP 28
#define P2_EXT_FLOW_NOHOLES 29
#define P2_EXT_FLOW_FRAME 30
#define P2_EXT_FLOW_LINE 31
#define P2_COMPR_RTP_HEADERS 29

#define BUGGY_FILETIME_OF_1970_01_01_H 27111902
#define BUGGY_FILETIME_OF_1970_01_01_L 3577643008ull

#define RANGE_MASK      0x1ffffffff
#define RANGE_RTP        0x80
#define RANGE_FRAMES     0xff
#define RANGE_DCH        0x10000
#define RANGE_MAX        0x100000

#define TYPE_CN_TERMINATED 0x120
#define FILETIME_TO_TIMEVAL_SKEW 0x19db1ded53e8000ull

enum service_states {
    SS_EVENT_FIFO = 2,
    SS_LISTENING = 3,
    SS_WAIT_FOR_FIRST_PH_CONN = 4,
    SS_WAIT_FOR_CONTROL_PH_CONN = 5,
    SS_WAIT_FOR_DATA_PH_CONN = 6,
    SS_BOTH_PH_CONNS = 7
};

typedef enum
{
    IPC_DATA = 0,
    IPC_CONNECT,
    IPC_ACCEPT,

```

```

        IPC_REJECT,
        IPC_DISCONNECT,
        IPC_REMOTE_ADDRESS,
        IPC_INVALID_MESSAGE
    } ipc_hdr_type;

typedef enum
{
    ROOT_CONNECTION = 0,
    DATA_CONNECTION,
    EVENT_CONNECTION,
    _LAST_CONNECTION
} ipc_connection_type;

typedef struct
{
    uint32_t    low;
    uint32_t    high;
} file_time;

#pragma pack(push, 1)
struct hook_state
{
#ifdef BYTE_ORDER == BIG_ENDIAN
    uint8_t hook;
    uint8_t direction;
    uint8_t len;
    uint8_t type;
#else
    uint8_t type;
    uint8_t len;
    uint8_t direction;
    uint8_t hook;
#endif
};

typedef struct p2_msg
{
    uint16_t opcode;
    uint16_t version;
    union {
        uint32_t context;
        uint32_t session;
    };
    file_time timestamp;
    union {
        uint32_t flow;
        uint32_t flows;
        uint32_t param0;
    };

    union {
        uint32_t compression;
        struct hook_state hs;
        struct {
#ifdef BYTE_ORDER == BIG_ENDIAN
            uint16_t offs;
            uint8_t len;
            uint8_t type;
#else
            uint8_t type;

```



```

        uint8_t len;
        uint16_t offs;
#endif
        } info;
        struct {
#if BYTE_ORDER == BIG_ENDIAN
            uint16_t slot;
            uint16_t type;
#else
            uint16_t type;
            uint16_t slot;
#endif
        } target;
        uint32_t id;
        uint32_t param1;
    };
    union {
        uint32_t flow_len;
        uint32_t param2;
        uint32_t size;
    };
} p2_msg;
#pragma pack(pop)

typedef enum
{
    P2_MSG_OK = 0,
    P2_MSG_FAILED,
    P2_UNSUPPORTED_VERSION,
    P2_INVALID_MSG,
    P2_INVALID_COMPRESSION,
    P2_INVALID_FLOW,
    P2_MSG_LAST,
} p2_msg_opcodes;

typedef enum
{
    P2_CC_EVENT_CALLER_ID = P2_MSG_LAST,
    P2_CC_EVENT_KEY_ID,
    P2_CC_EVENT_HOOK_STATE,
    P2_CC_EVENT_FLOW_STATE,
    P2_CC_EVENT_VOX_STATE,
    P2_CC_FLOW_STATUS_GET,
    P2_CC_FLOW_STATUS,
    P2_CC_TARGET_SET,
    P2_CC_TARGET_DEL,
    P2_CC_EVENT_HOOK_STATE_XX42_FEATURE,
    P2_CC_EVENT_DIR_STATE,
    P2_CC_EVENT_LOG_INFO,
} p2_msg_event_opcodes;

typedef enum
{
    P2_HOOK_ON = 0,
    P2_HOOK_OFF,
    P2_HOOK_UNKNOWN,
} p2_cc_hook_state;
typedef enum
{
    P2_VOX_ON,
    P2_VOX_OFF,

```



```

        P2_VOX_UNKNOWN,
    } p2_vox_state;
typedef enum
{
    P2_LINE_CONNECTED,
    P2_LINE_DISCONNECTED,
    P2_LINE_DISABLED,
    P2_LINE_UNKNOWN,
} p2_cc_line_state;
typedef enum
{
    P2_CALL_INCOMING,
    P2_CALL_OUTGOING,
    P2_CALL_UNKNOWN
} p2_cc_call_direction;

typedef enum
{
    P2_DATA_FLOW_START = P2_MSG_LAST,
    P2_DATA_FLOW_STOP,
    P2_DATA_FLOW,
} p2_msg_data_opcodes;

typedef enum
{
    P2_INFO_SINGLE = 0,
    P2_INFO_COMPLEX,
    P2_INFO_COMPLEX_XY,
} p2_cc_info_type;

typedef struct s_p2_hook_descr {
    uint32_t flow;
    uint16_t task;
    uint8_t type;
    uint8_t dir;
} p2_hook_descr;

typedef enum {
    P2_FLOW_UNKNOWN = 0,
    P2_FLOW_AUDIO,
    P2_FLOW_VIDEO,
    P2_FLOW_FILE,
    P2_FLOW_H225,
    P2_FLOW_H235,
    P2_FLOW_H245,
    P2_FLOW_T38,
    P2_FLOW_T120,
} p2_cc_flow_type;

typedef enum {
    P2_TARGET_ATTR_LOGIN = 0x0,
    P2_TARGET_ATTR_PHONE_NUMBER = 0x02,
    P2_TARGET_ATTR_IP = 0x03,
    P2_TARGET_ATTR_EMAIL = 0x04,
    P2_TARGET_ATTR_IP_RANGE = 0x07,
    P2_TARGET_ATTR_IP_MASK = 0x08,
    P2_TARGET_ATTR_UNI_LOGIN = 0x09,
    P2_TARGET_ATTR_UNI_ALIAS = 0x0A,
    P2_TARGET_ATTR_MAC = 0x0B,
    P2_TARGET_ATTR_IGN_BASE = 0x1000,
    P2_TARGET_ATTR_MODE = 0x1000,

```

```

        P2_TARGET_ATTR_PRIORITY          = 0x1001,
    } p2_target_attr;

#pragma pack(push, 1)
typedef struct dr_attr_ph {
    uint32_t    req_uid;    /* request unique id          */
    uint64_t    r_proto;    /* requested protos           */
    uint16_t    attr_cnt;   /* attributes count in request */
    unsigned char res0;     /* reserved                   */
    unsigned char res;      /* reserved                   */
} dr_attr_ph; /* 16 */

typedef struct dso_attr_dh {
    uint32_t    attr_id;
    uint16_t    attr_len;
    uint16_t    attr_state; /* attribute status.
                           0 - closed,
                           1 - opened,
                           2 - not affected,
                           3 - replace all
                           */
} dso_attr_dh; /* 8 */

typedef struct {
    uint8_t type;
    uint16_t src_id;
    uint16_t dst_id;
    uint32_t size;
    uint8_t data[0];
} __attribute__((packed)) ipc_msg;

#pragma pack(pop)

#endif /* __IPCGLUE_H */

```