Итак. Мы закончили с простыми текстовыми форматами, которые имеют популярность в интернете. Теперь давайте перейдём к хардкорным форматам. Следующей формат относится уже, непосредственно, не к популярности интернета, а к популярности внутри экосистемы go и всех сервисах, написанных на этом языке.

Соответственно, формат gob, Go Binary, (НРЗ 00:23). Соответственно, что это такое? Это бинарный протокол. Это что значит? Это значит, что данные представлены не в виде текста какого-то, не в виде потока, значит, соответственно, байтов, представляющих какую-то строку, а именно в виде потока байтов, которые представляют поток байтов. То есть — это протокол, который стоит на уровень ниже текста. Он не преобразуют данные в текстовые значения, он как-то форматирует данные, чтобы они помещались уже, непосредственно, в просто поток байтов. И как тексты, они уже не будут иметь такого смысла. Соответственно, поскольку — это бинарный протокол, не нужно применять операции сравнения строк, не нужно применять операции, соответственно, преобразование в строку, не нужно форматировать числа в строку.

Работает куда быстрее, чем текстовые протоколы. Как раз из-за всего отсутствия накладного уровня. Но не может читаться людьми без помощи специального софта. Не можете вы взять строку байтов, вернее, вы можете, конечно, его попробовать прочитать, но это намного сложнее, чем взять json, и понять, что там происходит.

И пока работает только для приложений на golang-е и приложений на C. Есть библиотека отдельная, которая работает с форматом Go Binary. Вследствие чего, применяется он только пока для разработки микросервисов и какого-то обмена данных между ними. То есть, когда у вас есть целая инфраструктура микросервисов и все они написаны на go, Go Binary — это очень хороший формат,

потому что он встроен из коробки, он неприхотлив, он быстрый, и вы можете прямо сесть и поехать.

Давайте посмотрим, как как раз на него сесть и поехать.

Итак, переходим к бинарных форматам. Формат gob, Go Binary. Здесь, в принципе, всё будет несколько отличаться от json-ов, yaml-ов, xml-ей. И не настолько это выглядит прикольно. То есть как происходит кодирование того же самого gob-а? Мы создаём buffer, байтовый buffer. Мы создаём, значит, encoder — это штука, которая, собственно говоря, кодирует представления в битовый формат. Например, для примера я взял map-у. Вы можете взять любую структуру. То есть смысл формата Go Binary в том, что вы взяли структуру, её как-то назвали, у неё как-то названы поля, вы её передаёте. Имя структуры не сохраняется, сохраняются поля, типы этих полей, и они вот как раз у вас записываются на (НРЗ 02:59).

То есть на другой стороне вы можете принять, перекодировать их в какую-то другую структуру. Пока у вас названия полей такие же, всё будет работать. Соответственно, нельзя передавать нулевые поля, указатели которого у вас есть в структуре будут сглажены и значения будут переданы. И, собственно говоря, циклические типы, когда тип…, (НРЗ 03:23), вернее типы, когда тип сам на себя ссылается. Или, когда есть ссылка внутри этого типа, в принципе, могут поддерживаться. Но надо быть аккуратными.

Здесь просто я показываю пример. То есть показываю. Например, вот есть encoder. Если он не вернул ошибку, значит в buffer-е у нас теперь будет лежать байтовое значение этого самого, gob. И в декодированном виде мы уже получим, соответственно, наш массивчик.

Давайте поставим точки отладки на оба. И посмотрим, значит, что у нас лежит в buffer-е. В buffer-е у нас будет какая-то битовая репрезентация нашего типа map. И соответственно, в следующем, в декодированном формате мы получаем эти самые байты из input-а, создаём новый buffer, создаём encoder. И просто делаем все в обратном порядке.

Да, здесь мы уже знаем, какая структура нам нужна — это важно, потому что мы должны представить такую структуру. В принципе, здесь то же самое, что у нас в json-е. И уже мы декодирует. Почему это не анмаршалинг, потому что маршалинг — это представление чего-то в текстовой форме. Анмаршалинг, соответственно, представление обратно в исходную форму. Соответственно, декодирование не является анмаршалингом, потому что текстовой формулы у нас нет. Соответственно, здесь у нас в «m», соответственно, сохранилась вот такая вот структура.

В принципе, здесь больше давать примеров какого-то смысла нет — это очень простой такой прям (HP3 04:59) путь в go для того, чтобы представлять структуры и пересылать их в другом языке. В жизни Ops-а вы с этим протоколом столкнетесь, ну достаточно редко пока на данный момент.