ПРИВАТНОСТЬ

Предбиллинг. Разбираемся, как мобильные операторы хранят и обрабатывают наши данные

Дмитрий Кузьмин , 20 минут назад 2 мин на чтение 0 0 234

Мобильная версия статьи

Содержание статьи

Мобильные операторы получают массу данных и метаданных, по которым можно узнать очень многое о жизни отдельно взятого абонента. А поняв, как обрабатываются и как хранятся эти данные, ты сможешь отследить всю цепочку прохождения информации от звонка до списания денег. Если же говорить о модели внутреннего нарушителя, то здесь возможности и подавно огромные, ведь защита данных вообще не входит в задачи систем предбиллинга.

Для начала нужно учитывать, что абонентский трафик в сети телеком-оператора генерируется и поступает с разного оборудования. Это оборудование может формировать файлы с записями (файлы CDR, логи RADIUS, текст в ASCII) и работать по разным протоколам (NetFlow, SNMP, SOAP). И нужно контролировать весь этот веселый и недружный хоровод, снимать данные, обрабатывать и передавать дальше в биллинговую систему в формате, который будет предварительно стандартизован.

При этом везде бегают абонентские данные, доступ к которым желательно не предоставлять посторонним. Насколько защищена информация в такой системе с учетом всех цепочек? Давай разбираться.

Зачем предбиллинг телеком-операторам?

Считается, что абоненты хотят получать все более новые и современные виды услуг, но нельзя постоянно менять для этого оборудование. Поэтому реализацией новых услуг и способами их предоставления должен заниматься предбиллинг — это его первая задача. Вторая — анализ трафика, проверка его корректности, полноты загрузки в абонентский биллинг, подготовка данных для биллинга.

С помощью предбиллинга реализованы различные сверки и дозагрузки данных. Например, сверка состояния услуг на оборудовании и в биллинге. Бывает, абонент пользуется услугами при том, что в биллинге он уже заблокирован. Либо он пользовался услугами, но с оборудования не поступили записи об этом. Ситуаций может быть множество, большинство таких моментов и решается с помощью предбиллинга.

Когда-то я писал курсовую работу по оптимизации бизнес-процессов компании и расчету ROI. Проблема с расчетом ROI была не в том, что не было исходных данных, — я не понимал, какой «линейкой» их мерить. Примерно так же часто бывает с предбиллингом. Можно бесконечно настраивать и улучшать обработку, но всегда в какой-то момент обстоятельства и данные сложатся так, что произойдет исключение. Можно идеально выстроить систему работы и мониторинга вспомогательных систем

биллинга и предбиллинга, но невозможно обеспечить бесперебойную работу оборудования и каналов передачи данных.

Поэтому и существует дублирующая система, которая занимается проверкой данных в биллинге и данных, ушедших от предбиллинга в биллинг. Ее задача — поймать то, что ушло с оборудования, но по какой-то причине «не легло на абонента». Эту роль дублирующей и контролирующей предбиллинг системы обычно играет FMS — Fraud Management System. Конечно, ее основное предназначение — вовсе не контроль предбиллинга, а выявление мошеннических схем и, как следствие, мониторинг потерь и расхождений данных с оборудования и биллинговых данных.

На самом деле вариантов использования предбиллинга очень много. Например, это может быть выполнение сверки между состоянием абонента на оборудовании и в CRM. Такая схема может выглядеть следующим образом.

1. С помощью предбиллинга по SOAP получаем данные с оборудования (HSS, VLR, HLR, AUC, EIR).
2. Преобразуем исходные RAW-данные в нужный формат.
3. Делаем запрос в смежные системы CRM (базы данных, программные интерфейсы).
4. Производим сверку данных.
5. Формируем записи-исключения.
6. Делаем запрос в систему CRM на синхронизацию данных.
7. Итог — абонент, качающий фильм в роуминге в ЮАР, блокируется с нулевым балансом и не уходит в дикий минус.

Еще один пример использования — накопление данных и дальнейшая их обработка. Такой вариант возможен, когда у нас тысячи записей с оборудования (GGSN-SGSN, телефония): выбрасывать все эти записи в детализацию абонента — полнейшее безумие, не говоря уже о том, что мы адски нагружаем все системы таким количеством мелких данных. По этой причине подойдет следующая схема, которая разрешает проблему.

1. Получение данных с оборудования.
2. Агрегация данных на предбиллинге (ждем, когда соберутся все нужные записи по какому-либо условию).
3. Отправка данных в конечный биллинг.
4. Итог — вместо 10 тысяч записей мы отправили одну с агрегирующим значением счетчика потребленного интернет-трафика. Сделали всего один запрос к базе данных и сэкономили кучу ресурсов, включая электричество!

Это всего лишь типовые схемы работы. Формат статьи не позволяет привести примеры более сложных схем (например, Big Data), но они тоже встречаются.

Как усмирить зоопарк?

Чтобы было понятнее, как это работает и где здесь могут возникнуть проблемы, давай возьмем систему предбиллинга Hewlett-Packard Internet Usage Manager (HP IUM, в обновленном варианте eIUM) и на ее примере посмотрим, как работает подобный софт.

Представь большую мясорубку, в которую бросают мясо, овощи, буханки хлеба — все, что только можно. То есть на входе самые разные продукты, но на выходе все они приобретают одинаковую форму. Мы можем поменять решетку и получим на выходе другую форму, но принцип и путь обработки наших продуктов останется прежний — шнек, нож, решетка. Это и есть классическая схема предбиллинга: сбор, обработка и вывод данных. В предбиллинге IUM звенья этой цепочки называются encapsulator, aggregator и datastore.

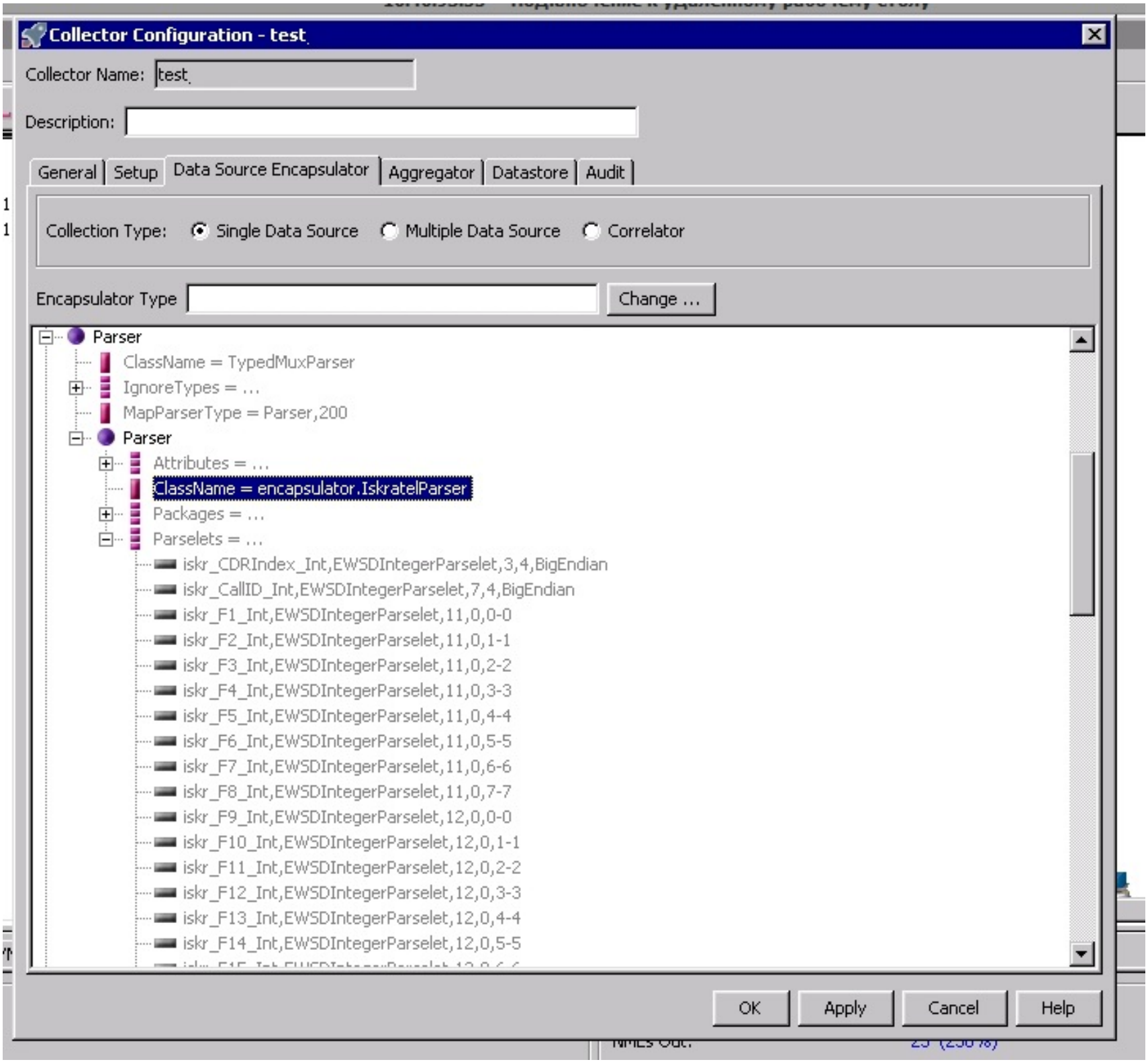
Тут необходимо понимать, что на входе у нас должна присутствовать полнота данных — некий минимальный объем информации, без которого дальнейшая обработка бесполезна. При отсутствии какого-то блока или элемента данных мы получаем ошибку или предупреждение, что обработка невозможна, так как операции не могут быть выполнены без этих данных.

Поэтому очень важно, чтобы оборудование формировало файлы-записи, которые имели бы строго определенный и установленный производителем набор и тип данных. Каждый тип оборудования — отдельный обработчик (коллектор), который работает только со своим форматом входных данных. Например, нельзя просто так взять и закинуть файл с оборудования CISCO PGW-SGW с интернет-трафиком мобильных абонентов на коллектор, который обрабатывает поток с оборудования фиксированной связи Iskratel Si3000.

Если мы так сделаем, то в лучшем случае получим исключение при обработке, а в худшем у нас встанет вся обработка конкретного потока, так как обработчик-коллектор упадет с ошибкой и будет ждать, пока мы не решим проблему с «битым» с его точки зрения файлом. Здесь можно заметить, что все системы предбиллинга, как правило, критично воспринимают данные, на обработку которых не был настроен конкретный обработчик-коллектор.

Изначально поток разобранных данных (RAW) формируется на уровне энкапсулятора и уже здесь же может быть подвергнут преобразованиям и фильтрации. Так делается, если нужно до схемы агрегации произвести с потоком изменения, которые должны быть в дальнейшем применены ко всему потоку данных (когда он будет проходить через

различные схемы агрегации).



Файлы (.cdr, .log и прочие) с записями об активности пользователей-абонентов поступают как с локальных источников, так и с удаленных (FTP, SFTP), возможны варианты работы и по другим протоколам. Разбирает файлы парсер, с помощью разных классов Java.

Так как система предбиллинга в нормальном режиме работы не предназначена для хранения истории обрабатываемых файлов (а их может быть сотни тысяч в сутки), то после обработки файл на источнике удаляется. По разным причинам файл не всегда может быть удален корректно. В результате бывает, что записи из файла обрабатываются повторно или с большим опозданием (когда удалить файл получилось). Для предотвращения таких дублей существуют механизмы защиты: проверка на дубли файлов или записей, проверка на время в записях и прочее.

Одно из самых уязвимых мест здесь — это критичность к размеру данных. Чем больше мы храним данных (в памяти, в базах данных), тем медленнее мы обрабатываем новые данные, тем больше мы потребляем ресурсов и в итоге все равно достигаем предела, после которого вынуждены удалить старые данные. Таким образом, для хранения этих метаданных обычно используются вспомогательные БД (MySQL, TimesTen, Oracle и так далее). Соответственно, получаем еще одну систему, которая влияет на работу предбиллинга с вытекающими вопросами безопасности.

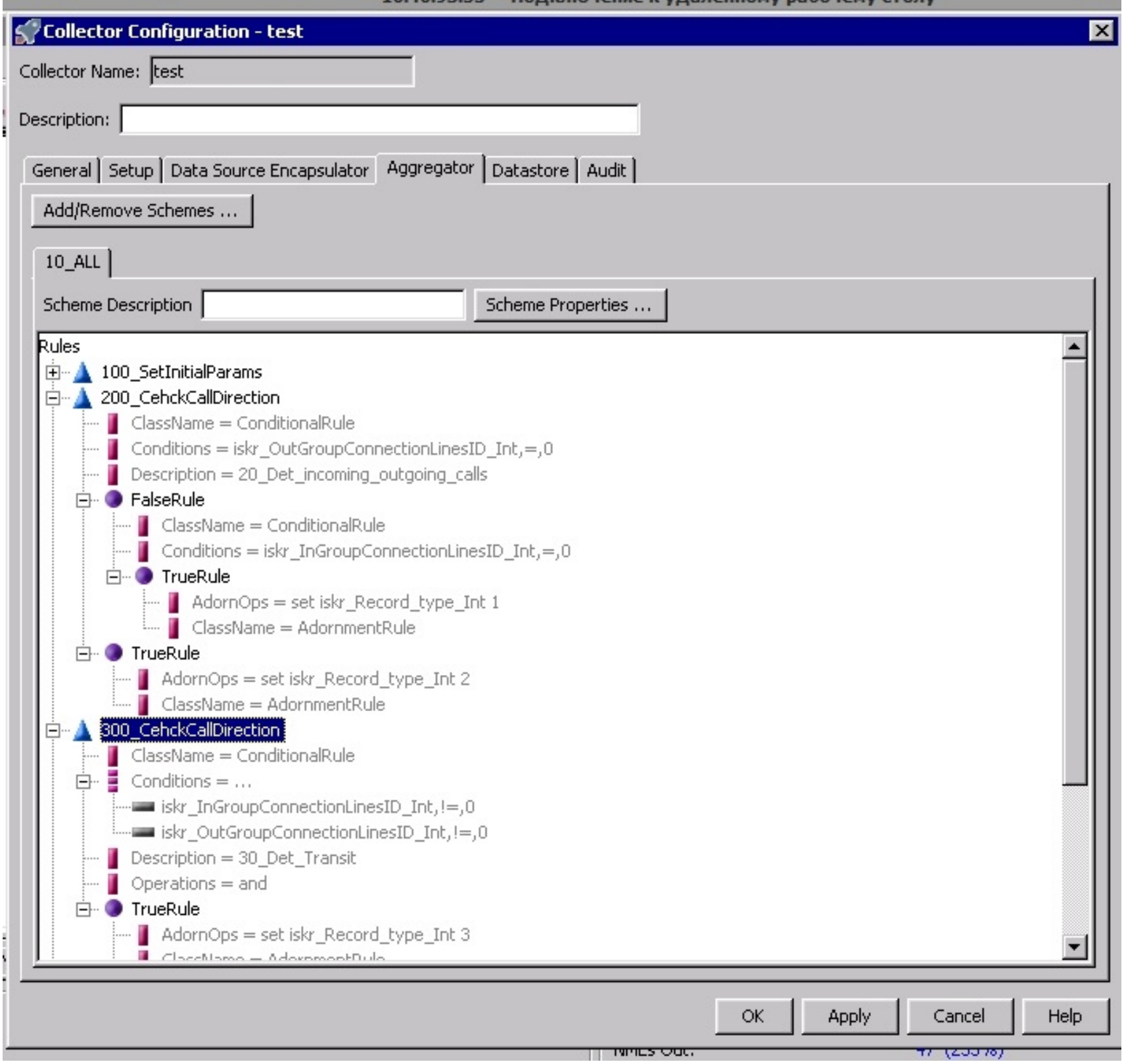
Что в черном ящике?

Когда-то на заре подобных систем использовались языки, которые позволяли эффективно работать с регулярными выражениями, — таким, например, был Perl. Фактически почти весь предбиллинг, если не брать во внимание работу с внешними системами, — это правила разбора-преобразования строк. Естественно, лучше регулярных выражений тут ничего не найти. Постоянно растущий объем данных и повышение критичности к времени вывода новой услуги на рынок сделали применение таких систем невозможным, так как тестирование и внесение изменений занимало много времени, масштабируемость была низкой.

Современный предбиллинг — это набор модулей, как правило написанных на Java, которыми можно управлять в графическом интерфейсе с помощью стандартных операций копирования, вставки, перемещения, перетаскивания. Работа в этом интерфейсе проста и понятна.

Для работы в основном используется операционная система на базе Linux или Unix, реже — Windows.

Основные проблемы обычно связаны с процессом тестирования или выявления ошибок, так как данные проходят по множеству цепочек правил и обогащаются данными из других систем. Видеть, что происходит с ними на каждой стадии, не всегда удобно и понятно. Поэтому приходится искать причину, отлавливая изменения нужных переменных при помощи логов.



Слабость этой системы — ее сложность и человеческий фактор. Любое исключение провоцирует потерю данных или неправильное их формирование.

Обрабатываются данные последовательно. Если на входе у нас ошибка-исключение, которая не позволяет корректно принять и обработать данные, встает весь входной поток либо порция некорректных данных отбрасывается. Разобранный RAW-поток поступает на следующую стадию — агрегацию. Схем агрегации может быть несколько, и они изолированы друг от друга. Как если единый поток воды, поступающий в душ, пройдя через решетку лейки, разделится на разные потоки — одни толстые, другие совсем тонкие.

После агрегации данные готовы к доставке потребителю. Доставка может идти как напрямую в базы данных, так и записью в файл и отправкой его дальше либо просто записью в хранилище предбиллинга, где они будут лежать, пока его не опустошат.

После обработки на первом уровне данные могут передаваться на второй и далее. Такая лестница необходима для увеличения скорости обработки и распределения нагрузки. На второй стадии к нашему потоку данных может добавляться другой поток, смешиваться, делиться, копироваться, объединяться и так далее. Конечная стадия — это всегда доставка данных в системы, которые его потребляют.

В задачи предбиллинга не входит (и это правильно!):

- мониторить, поступили и доставлены ли входные-выходные данные, — этим должны заниматься отдельные системы;
- шифровать данные на любой стадии.

Далеко не весь поток поступающих данных подвергается обработке. Обрабатываются только те данные, которые нужны для работы. Тратить время на остальные нет смысла до того момента, пока они не понадобятся. Таким образом, из RAW-потока нужно брать только то, что нужно для схем агрегации. Из RAW (текстовые файлы, результаты запросов, бинарные файлы) парсится только необходимое.

Приватность

Здесь у нас полный расколбас! Начнем с того, что в задачи предбиллинга не входит защита данных в принципе. Разграничение доступа к предбиллингу нужно и возможно на разных уровнях (интерфейс управления, операционная система), но если мы заставим его заниматься шифрованием данных, то сложность и время обработки настолько увеличатся, что это будет совершенно неприемлемо и непригодно для работы биллинга.

Зачастую время от использования услуги до отображения этого факта в биллинге не должно превышать нескольких минут. Как правило, метаданные, которые нужны для обработки конкретной порции данных, хранятся в БД (MySQL, Oracle, Solid). Входные и выходные данные практически всегда лежат в директории конкретного потока-коллектора. Поэтому доступ к ним может иметь любой, кому он разрешен (например, root-пользователь).

Сама конфигурация предбиллинга с набором правил, сведениях о доступах к базам данных, FTP и прочему хранится в зашифрованном виде в файловой базе данных. Если неизвестен логин-пароль для доступа в предбиллинг, то выгрузить конфигурацию не так просто.

Любое внесение изменений в логику обработки (правила) фиксируется в лог-файл конфигурации предбиллинга (кто, когда и что менял).

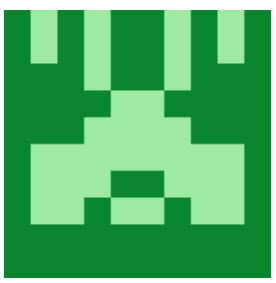
Даже если внутри предбиллинга данные передаются по цепочкам обработчиков-коллекторов напрямую (минуя выгрузку в файл), данные все равно временно хранятся в виде файла в директории обработчика, и при желании к нему можно получить доступ.

Данные, которые проходят обработку на предбиллинге, обезличены: они не содержат ФИО, адресов и паспортных данных. Поэтому даже если ты получишь доступ к этой информации, то персональных данных абонента отсюда не узнать. Зато можно поймать какую-то инфу по конкретному номеру, IP либо другому идентификатору.

Имея доступ к конфигурации предбиллинга, ты получаешь данные для доступа ко всем смежным системам, с которыми он работает. Как правило, доступ к ним ограничен непосредственно с сервера, на котором работает предбиллинг, но так бывает не всегда.

Если ты доберешься до директорий, где хранятся файловые данные обработчиков, то сможешь вносить изменения в эти файлы, которые ждут своей отправки потребителям. Часто это самые обычные текстовые документы. Тогда картина такая: предбиллинг данные принял и обработал, но в конечную систему они не пришли — пропали в «черной дыре».

И выяснить причину этих потерь будет сложно, так как потеряна только часть данных. В любом случае эмулировать потерю будет невозможно при дальнейшем поиске причин. Можно посмотреть данные на входе и выходе, но понять, куда они делись, будет невозможно. Злоумышленнику при этом остается только замести следы в операционной системе.



Дмитрий Кузьмин

Теги: Выбор редактора | Информационная безопасность | Мобильная связь | Персональные данные | сотовая связь | Статьи | Утечка данных