Tiger Shore
Management
Limited

Implementation the OpenRate Processing Pipelines for Ergatel

17th November 2008

Draft

Table of Contents

Document Control

Revisions

Version	Date	Description	Author	Approved
0.1a	17th Nov 2008	Draft version of the GSM Rating Pipeline	Ian Sparkes	

SeaSatCom Data Processing Architecture

This section describes the data processing architecture that is common to all processing pipelines. Where there are exceptions to these rules, they will be described in the sections on the individual pipelines.

In general, the SeaSatCom processing architecture is based around relatively few high value customers, producing relatively few events. For the purposes of planning at this time, the following key sizing attributes have been determined:

- There are currently around 5 tariff plans, but this number is quite dynamic
- There are currently around 100 active customers.
- Each customer produces an maximum of around 300 events per month

Event Processing Architecture

This section, which describes the overall data processing architecture is to be completed during the design phase of the project. This section is therefore a place holder.

Event Reprocessing

An important aspect of any processing architecture is the way that errors are handled. In the processing pipeline, reprocessing can occur for one of several reasons:

A) Due to a configuration error in the pipeline configuration (e.g. a band was missing, or a price list was wrong)

B) Due to an internal temporal problem (e.g. the customer is provisioned on the technical network and has created usage, but the customer account is not yet provisioned in the system)

C) Due to an external temporal problem (e.g. the customer has moved, but has failed to inform Ergatel of the change of address)

D) Due to a problem in upstream systems (e.g. mediation provided incorrect information)

It is expected that error handling will be performed using a “recycle/writeoff” processing method, that is, error records will be sent to a recycle file, and will be recycled until they meet the criteria for being written off. The write off criteria must be defined.

OpenRate Processing Architecture

OpenRate is a transactional batch processing system, which transports records for processing down a conceptual “pipeline”, made up of processing stages, each of which perform a discrete transformation on the records. As there can be many processing stages, the OpenRate architecture is multi-threaded, with a thread dedicated to each of the processing stages. Because telco processing is often processing intensive, this configuration gives us two distinct advantages:

- Full utilization of available hardware can easily be achieved, meaning that the hardware that is available is used to it's full potential
- Each record is processed in a “serial-parallel” manner (i.e. each discrete step is strictly serial, but many discrete steps may be in parallel). This means that contention issues are virtually eliminated, even when many threads are being applied to the processing.

Each processing pipeline has a single input (in this case an input table) which is polled at a configurable rate to see if new records have been found, an unlimited number of processing steps, and one or more outputs (either tables or files). After processing, the input records are excluded from future processing (either by deleting or marking them), and corresponding output records are persisted to downstream storage.

A transaction is created around a batch of records that has been found at the input during the polling, and each transformation must finish successfully before the overall transaction can be committed.

Event Input

The input records will be presented to the pipeline as files in an input location. After processing, the files will be renamed (OpenRate never cancels input information) for archiving.

As there are potentially three types of input information, it must still be decided if the processing should be done in three separate pipelines, or a single one that can handle all traffic types.

GSM Usage Pipeline

Introduction

The OpenRate rating pipeline for GSM telecommunication event records (“CDRs”) processes the incoming telecommunications traffic events in preparation for billing. The OpenRate pipeline accepts records from the SeaSatCom wholesaler, and passes the records that it processes to the OpenBRM billing system, which is based on jbilling. It therefore acts as a pure “matching” and “rating” component in the overall system architecture.

The telecommunications traffic that is to be processed arrives from a number of different network components. This traffic must be rated and placed on the appropriate customer invoice, with OpenRate providing the allocation and pricing capability.

Thus, the OpenRate pipeline performs the following tasks, which will be described more fully in the remainder of this document:

- “Matching”
    - Identifying the user who effected the traffic
    - Identifying the type of traffic
    - Identifying the tariff plans to be applied to the traffic
    - Identifying the “zone band” (which maps the concept of the distance of the call) to be applied to the traffic
    - Identifying the “time band” (which maps the concept of peak and off-peak calling) to be applied to the traffic
- “Rating”
    - Performing the calculation of the monetary impacts of the call
    - Rounding and formatting of the information in order that it can be easily accepted by the billing system
    - Enrichment of the traffic record for both the allocation to the customer account in jbilling, and the archiving of the event for reconciliation and auditing purposes.

High Level Design – GSM Usage Processing Pipeline

This section is intended to give an overview of the way that the traffic processing pipeline will locate and present the results to the downstream systems.

Module List

This section lists the modules used in the GSM traffic rating pipeline, with functional level information about the module processing.

Input Processing

The input to the traffic processing pipeline is performed by reading an input table:

Module Name	*Based On*	*Description*
GSMInputAdapter	FlatFileInputAdpater	This module reads the files that arrive in the input directory, and parses the data, preparing it for processing. The processing follows the standard OpenRate steps: 1) Scan the input directory for unrated CDRs for processing 2) Mark the unrated CDRs as being "in processing", by renaming the file 3) Process the CDRs 4) Mark the CDRs as "done", by moving the file to the "done" (transaction committed) or "error" (transaction rolled back) directory The input information will be presented to the processing pipeline as a stream of "SeaSatRecords", after performing the necessary mapping of the input information. The "SeaSatRecords" record will be based on the "RUMRating" record ancestor type, and will be extended according to the needs identified in this document. In order to distinguish the GSM records from the SAT records, the GSM records are given a RECORD_TYPE value of 20.

At this point, we have a stream of records formatted accordingly for the matching and rating.

Matching

The matching section of the pipeline recognises the attributes received in the input record, and compares them against the configuration to recover:

- The customer identity
- The zone model, time model and price model to use

The processing is performed by the following modules:

Module Name	*Based On*	*Description*
NormLookupGSM	RegexMatch	Normalisation - Perform normalisation based on the regular expression rules module to allow table driven maintenance of the normalisation rules. The fields “Destination” and “B_Number” are used to drive the normalisation. The module uses the configuration in the sheet “*Normalisation Map*” to perform the processing. **Cache Used:** NormCache **Loaded From Configuration Table:** NORM_MAP
OriginLookupGSM	AbstractRegexMatch	This module determines the location that the account holder was in at the time of the usage. This uses the value of the input record field “Origin” and maps it to the location. We use the “Libelle Facture” field to know the origin of the call. In the case that this is not filled, or filled with a “-” we map this is to mean that the call origin was “NATIONAL”. If the call has been marked as a “Passport” call, the origin of the call will be chosen as a passport zone, otherwise the zone will be a “MondePlus” zone. The passport status of the call is found by looking for the text “passport” in the CALL_TYPE field”. The module uses the configuration in the sheet “*Origin Destination Zone Map*” to perform the processing. **Cache Used:** OriginLookupCache **Loaded From Configuration Table:** ORIGIN_MAP
GSMDestinationLookup	AbstractRegexMatch	In order to know the destination of the call, we perform a number analysis on the normalised other party number, which was created by the Normalisation module. For this module we must distinguish between the cases that we have an originating call or a terminating call. For the case of a terminating call, we do not have the destination available, and it will be set to “None”. (Note: for processing purposes, we invert the origin and destination in the case of a terminating call. This allows the configuration to be more simple). If the call has been marked as a “Passport” call, the origin of the call will be chosen as a passport zone, otherwise the zone will be a “MondePlus” zone. The passport status of the call is found by looking for the text “passport” in the CALL_TYPE field”. To do this, we use the Normalised_B_Number as the input, and we use the configuration in the sheet “*Origin Destination Zone Map*” to perform the processing. **Cache Used:** GSMDestLookupCache **Loaded From Configuration Table:** GSM_DESTINATION_MAP

Module Name	*Based On*	*Description*
CustomerTariffLookupGSM	AbstractJBCustomer Lookup	Looks up the Jbilling customer ID and gets the products that the customer has by accessing the Jbilling database. The customer ID is important because it allows us to allocate the usgae event to the customer account quickly in Jbilling. The Products are important because they determine the tariffing that will be used. Three sorts of products are located: • Base products (in Jbilling with a prefix of "BP:"). Only one of these is allowed on each customer. Customers with multiple base products will have their records rejected. • Options (in Jbilling with a prefix of "OP:"). A customer ^may have multiple options at the same time. • Packages (in Jbilling with a prefix of "PACKAGE" for GSM packages and "SPACKAGE" for Satellite packages). Each record must have a base product to be rated. Customers with no base product have their records rejected. **Cache Used:** JBCustomerCache **Loaded From Configuration Table:** purchase_order, order_line
DiscountLookupGSM	AbstractRegexMatch	Looks up discounts (bundles) that should be applied to the record. There are two ways that a bundle can be applied to a record: • By being given in an option: An option contains some bundled minutes. • By being attached to a base product. It is also possible to have a base product contain some bundled minutes. **Cache Used:** DiscountLookupCache **Loaded From Configuration Table:** DISCOUNT_MAP
FriendsFamily	AbstractRegexMatch	This module implements "Friends and Family" numbers by checking if the called destination number is one of the Friends and Family numbers which has been defined for the calling number. If it is one of the Friends and Family numbers, the entire call duration is discounted. There is no limit to the number of minutes which may be discounted this way. **Cache Used:** FFCache **Loaded From Configuration Table:** FRIENDS_FAMILY
PromoCalcPreRating	AbstractBalance HandlerPlugIn	This module identifies if bundled minutes are available for this call, and if so, reduces the number of minutes to be rated by the bundle amount available. Minutes which have been included in a bundle are not rated. If a call exhausts the bundle minutes, the portion which can be taken from the bundle is taken, and the rest is rated. This results in a partially discounted call. **Cache Used:** BalCache **Loaded From Configuration Table:** COUNTER_BALS

Module Name	*Based On*	*Description*
GSMPriceModelLookup	AbstractRegexMatch	This module looks up the price model to use based on the call scenario, based on the “Price Sheet”, the Origin Zone and the Destination Zone. The PriceSheet is calculated based on the CDR Type and the direction of the call. This leads us to the following price sheets: ● National Voice Call Outgoing ● National Voice Call Incoming ● International Voice Call Outgoing ● International Voice Call Incoming ● National SMS Outgoing ● National SMS Incoming ● International SMS Outgoing ● International SMS Incoming ● National MMS Outgoing ● National MMS Incoming ● International MMS Outgoing ● International MMS Incoming ● National Data ● International Data The result of the lookup is the price model (rating strategy) to use for the event. The module uses the configuration in the sheet “*Rate Plan Map*” to perform the processing. **Cache Used:** GSMPriceLookupCache **Loaded From Configuration Table:** GSM_PRICE_MAP
RateOverride	AbstractRegexMatch	This module applies a run on rate (“RoR”) after an option has been exausted. A run on rate is a price which will be applied when a bundle applies, but the bundle is exhausted. **Cache Used:** RateOverrideLookupCache **Loaded From Configuration Table:** RATEPLAN_OVERRIDE

Rating

The rating section of the pipeline recognises the attributes received in the input record, and compares them against the configuration to recover:

- The price of the call
- The category of the call (used for presentation purposes in jbilling)
- The customer account to be used

The processing is performed by the following modules:

Module Name	***Based On***	***Description***
RateLookup	AbstractRUMRateCalc	This module performs the rating for each charge packet that has been created, based on the Zone Result, Time Result and customer tariff for each packet. Note that there will be one charge packet for each segment (e.g. a call that crosses from peak to off-peak will therefore have two charge packets, one for the peak segment, and one for the off-peak segment. The results of the rating will be written into each charge packet, and will be rolled up later. **Cache Used:** RateCache **Loaded From Configuration Table:** PRICE_MODEL, RUM_MAP

Output

After the rating, the records must be prepared for output:

- Gather up impacts to write the output information
- Deal with error records
- Persist the records

The processing is performed by the following modules:

Module Name	*Based On*	*Description*
ChargeRollup	StubPlugin	This simple module will gather the results of the rating into a final “chargeable value” by rolling up and rounding the values found for each call segment.
SuspensePreparation	RegexMatch	This module checks the error status of each record, and based on whether the record has acquired an error during processing, and writes the output stream information. The output stream mapping will be: ● If the record has no error: jbillingOutput ● If the record has an error: SuspenseOutput This module will have a table driven configuration to allow the output streams to be defined as a configuration. **Cache Used:** SuspensePreparationCache **Loaded From Configuration Table:** SUSPENSE_MAP
Dump	Dump	This module writes the diagnostic information to the disk, configurable to write for all/none/errored records.
SuspenseOutput	FlatFileOutputAdapter	This module writes the output file of the records that will go to recycling with errors caused by an error in the configuration. 1) Create the output file 2) Spool the CDRs to the output file 3) Close the transaction, either committing (rename file) or rolling back (cancel file) as per the result of the processing transaction.
jbillingOutput	FlatFileOutputAdapter	This module writes the output file of the records that will go to jbilling. Only correctly rated CDRs go to this file. 1) Create the output file 2) Spool the CDRs to the output file 3) Close the transaction, either committing (rename file) or rolling back (cancel file) as per the result of the processing transaction.
SuspenseOutput	FlatFileOutputAdapter	This module writes the output file of the records that will go to recycling with errors caused by an unknown error. 1) Create the output file 2) Spool the CDRs to the output file 3) Close the transaction, either committing (rename file) or rolling back (cancel file) as per the result of the processing transaction.
RejectOutput	FlatFileOutputAdapter	This module writes the output file of the records that will be written off as no further recycling attempt will be made. 1) Create the output file 2) Spool the CDRs to the output file 3) Close the transaction, either committing (rename file) or rolling back (cancel file) as per the result of the processing transaction.
SuspenseOutput	FlatFileOutputAdapter	This module writes the output file of the records that will go to recycling with errors caused by an error in the Jbilling coinfiguration (e.g. product not found). 1) Create the output file 2) Spool the CDRs to the output file 3) Close the transaction, either committing (rename file) or rolling back (cancel file) as per the result of the processing transaction.

Module Name	*Based On*	*Description*
BalanceOutputAdapter	FlatFileOutputAdapter	Writes balance updates out to the balance output. These are picked up later and put into the COUNTER_BALS table, which keeps the database updated with the in memory status. The COUNTER_BALS table is only read at framework startup.