CERT: SMS Parsing

Data Model Blueprint: msg_workflow

Field Name	Purpose	Datatype/Table
source_task_id	Inbound Email/SMS Source (Another workflow in case of chained workflows).	FK referenced from scheduler_task.
workflow_task_id	Parsing workflow	FK referenced from scheduler_task.
s3.meta_fields()		metadata

Changes in msg_log:

Field Name	Purpose	Datatype
is_parsed	To process the log for parsing unparsed messages.	Boolean
source_task_id	Inbound Email/SMS Source	FK referenced from scheduler_task.
reply	To store the result of the 1st pass parser.	text

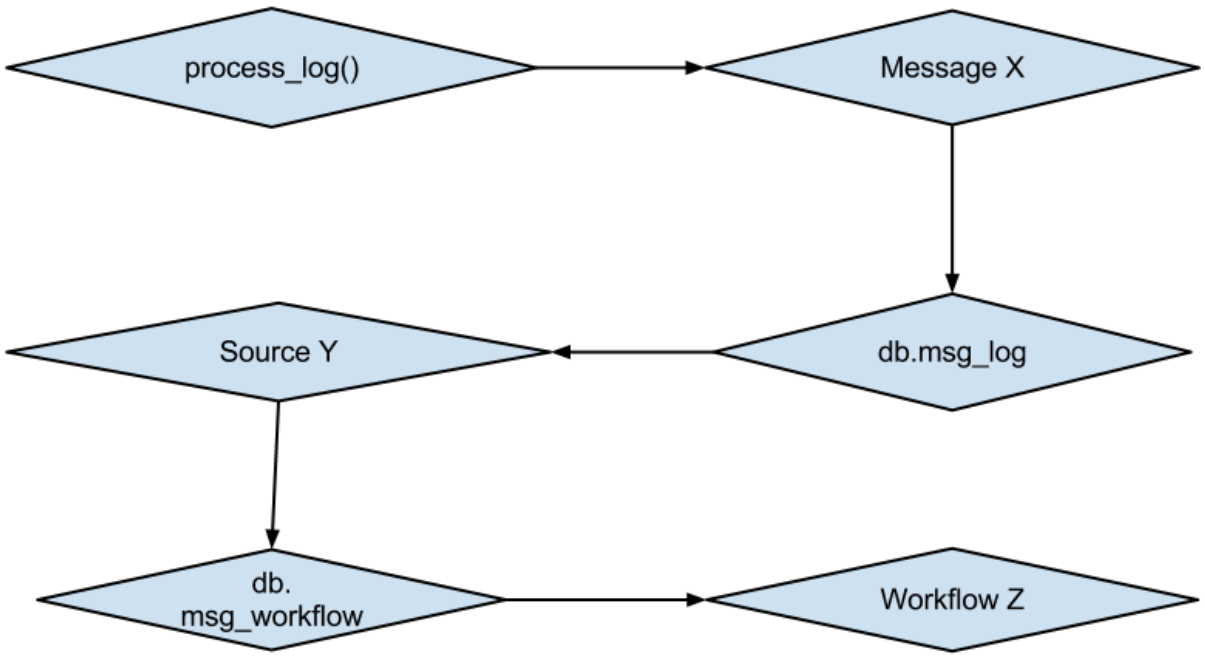
The msg_workflow table is described above in the table.Here,both “Source” and “Workflow” are Foreign Keys referenced from the scehduler_task table.These are scheduled tasks defining the incoming Email/SMS source/connection (inbound email/SMS handler : See https://github.com/flavour/eden/blob/master/models/tasks.py#L49) and the parsing workflow task (See below for the parsing scheduler task) respectively.
New records are inserted into the msg_workflow table when a new inbound message source is defined i.e. the workflow for that particular source and the source task itself are the DB entries/records.However, we would rather use prepopulate folders for this purpose;but using the above approach is a viable option. The parser task method takes both the fields as args, where it maps each source with the type of workflow required.
Data model changes and designing of process_log() has already been done to implement the current parse_message() routine.(See https://github.com/flavour/eden/pull/57).

UI/Prepop:

The data model is integrated with the prepopulate folders (or a sub-folder say private/prepopulate/parsing) which serves as the initial UI.The post-install UI will consist of a CRUD interface admin panel, a simple s3_rest_controller().However, eventually this is planned to be the part of the WebSetup.

Task Scheduler Details:

The parsing rules are defined in s3parsing.py.These are imported by tasks.py to define different parsing tasks/workflow.These tasks are instantiated in zzz_1st_run.py by calling the schedule_task() routine, say, process_log().The purpose of process_log() will be to parse the messages which have not been parsed yet.To identify the unparsed records in msg_log , a boolean valued field say “is_parsed” (or “is_processed” ?) is added.Now, the routines which defines the parsing rules (e.g. parse_message()) are scheduled as parsing workflows :workflow_task_id.Therefore, when the scheduler processes the log, it greps for the records/messages with ‘is_parsed’ set to False, and then it chains the concerned parsing task(this is achieved by the msg_workflow table, the ‘source_task_id’ field in msg_log will help retrieve the respective parsing workflow_task_id from msg_workflow).
Hence, the source is synchronised in both msg_workflow and msg_log.After,the message has been parsed or the apt action is taken (which may include generating a reply or directing the message to the concerned module), the respective message records in msg_log are flagged as “parsed” (or “processed” ?).
This has also been discussed in detail here:
https://docs.google.com/document/d/1tVZ3KJUp5ieFKCCJ_FosqsHIXqtsXErjQlrano-iDUE/edit?pli=1 .



The diagram below illustrates the purpose behind having a msg_workflow,how it operates and the possible relations between the different workflows.