Telephony Capture Service

Software Requirements Document

Version



Table of Contents

1 Introduction 3

1.1 Purpose 3

1.2 Scope 3

1.3 Definitions 4

1.4 References 4

1.5 Open Issues 4

2 Overall Description 5

2.1 Product Perspective 5

2.1.1 External Interfaces 5

2.1.2 Internal Interfaces 5

2.1.3 User Interface 5

2.2 Project Deliverables 5

2.3 Technical Constraints 6

2.4 User Characteristics 6

3 Specific Requirements 7

3.1 TCS Requirements 7

3.2 Mangle SMDR Files 7

3.3 Test Tool Requirements 8

3.3.1 PBX Simulator 8

3.3.2 Telecom Management System Simulator 9

3.4 Database Requirements 9

3.5 Deployment Requirements 10

3.6 Test Requirements 10

3.7 Performance Requirements 10

# Introduction

## Purpose

This document details the software requirements for a Telephony Capture Service (TSC) that receives Station Messaging Detail Records (SMDR) from a PBX and stores them to a database. The TCS is destined to replace an existing Telecom Management System (TMS), although a transition phase is anticipated in that the TCS can be configured to forward on all incoming data to the legacy TMS.

## Scope

Because the environment into which the TCS is to be installed is not currently running a server-side database, the scope of the project also includes the usual deliverables that one would expect with any database installation: installation, backup and recovery processes, and related technical documentation.

Other closely aligned support tools are the following:

* A routine that ingests datasets of SMDR records and creates corresponding datasets containing otherwise identical information, except that unknown phone numbers are mangled (scrambled). This utility provides a means to create versions of the SMDR data that can be released for off-site usage.
* Test utilities:
  1. A PBX simulator that forwards a realistic stream of data to the TCS. Note: during the first installation this same utility is also used to load the historical set of SMDR data into the database.
  2. A TMS simulator (accepts data from the TCS).

Database backups are to be delivered to the host platform on which Docker is running. These are managed as a circular rotation of ‘n’ backups, with the oldest being purged (the size of the circular backup queue and the backup schedule are both configurable).

It is assumed that any further off-platform recording of the backup files is to be provided by the existing backup environment.

## Definitions

**PBX**Private Branch Exchange

**SMDR**Station Messaging Detail Record. Its complete definition can be found in Appendix A.

**TMS**  
Telecom Management System

**TCS**  
Telephony Capture Service

## References

1. [SMDR Fields IPO 9.1.4 - required fields.docx](SMDR%20Fields%20IPO%209.1.4%20-%20required%20fields.docx)
2. [Telephony Capture Service Test Management Plan](TCS Test Management Plan.docx)

## Open Issues

1. Clarity is required concerning the stream of data coming from the PBX. There is more than SMRD messages flowing on this circuit. The current state of development is that SMDR messages are expected to be bracketed by a byte pattern (in hex): ‘00 02 00 00 00 00’, as the prefix, and ‘0d 0a’, as the suffix. Nevertheless, it is not known whether this simple message structure is unique to SMDR messages or not. Note: as an aside, all data bytes received from the PBX are passed on to the TMS, meaning that the TMS will continue to receive whatever data it was receiving before the installation of the TCS.
2. Investigate a means to safely identify duplicated SMDR data. This is required should a stream of SMDR messages need to be replayed.
3. The specific nature of the Docker installation is still under discussion in a couple of different senses:
   1. Docker to be installed within a Virtual Machine or installed on bare metal?
   2. Which host operating system is Docker to be installed into?
4. At the point of writing, an environment variable (see Appendix A), **DOCKER\_MACHINE\_IP**, needs to be configured manually. It is believed that this variable can be identified implicitly in software and therefore abandoned. This is being investigated.

# Overall Description

## Product Perspective

### External Interfaces

The TCS has 3 external interfaces, one input and two output:

1. An input stream of SMDR messages from the PBX.
2. An optional output interface to a legacy Telecom Management System (TMS).
3. An output interface to a relational database table whose columns parallel the CSV field content of SMDR messages.

### Internal Interfaces

The TCS receives SMDR messages from the PBX and must direct them both to the TMS and the database, but either or both of these may not be available and hence data destined for these output interfaces may be queued until such time that the TMS and / or the database are once again available. Further, queued data must persist should the TCS itself fail or is otherwise unavailable. The selected queuing service, [RabbitMQ](https://www.rabbitmq.com/), provides the required services.

### User Interface

The project’s User Interface components are minimal. These are:

1. A means to both view and edit a suite of environment variables that collectively describe the behavior of the TCS (port numbers, host addresses, passwords, switches, etc.).
2. A means to view the number of items in the two queues (should one or both of these queues start growing, this is indicative of a problem with the TMS and / or the database).
3. A means to edit various database configuration parameters and tables as required and according to project documentation.

## Project Deliverables

All of the following to be delivered via GitHub:

* NodeJS production source code (written in TypeScript)
* NodeJS test source code (written in TypeScript)
* Executable production JavaScript code (as transpiled from TypeScript)
* Executable test JavaScript code (as transpiled from TypeScript)
* Projects documents: The present document, the Test Management Plan, and any diagram files that have been generated to support technical or user documentation
* Various configuration files
* TCS application and deployment documentation
* Database installation documentation
* Database backup and restore documentation

## Technical Constraints

* Programming language: [TypeScript](https://www.typescriptlang.org/) 2.x
* Server Platform: [NodeJS](https://nodejs.org/en/) 7.x
* Database [PostgreSQL](https://www.postgresql.org/): 9.6.x
* Queue Message Broker: [RabbitMQ](https://www.rabbitmq.com/): 3.6.x
* Delivery Container: [Docker](https://www.docker.com/) 1.x

## User Characteristics

The users of the TCS are technical personnel who are familiar with the environment within which the TCS runs.

# Specific Requirements

## TCS Requirements

In parallel, the TCS shall:

* Receive TCP Segment Data from the PBX and optionally deliver this data unmodified to a persisted TMS\_QUEUE.

Note: If and when the Telecom Management System is deemed unnecessary, then the TCS flow to the TMS\_QUEUE can be disabled.

* The TCS shall receive data from the TMS\_QUEUE and forward it on to the TMS.
* The TCS shall receive TCP Segment Data from the PBX, isolate messages found between the data patterns (in hex) “00 02 00 00 00 00” and “0a 0d” (carriage return, line feed), and delivers such messages to the DATABASE\_QUEUE.
* The TCS shall receive SMDR messages from the DATABASE\_QUEUE, parse them into their respective CSV fields (SMDR messages are of a CSV format), and stores them into a database table, SMDR.

## Mangle SMDR Files

A mangling tool shall be provided which supports the following requirements:

* The tool shall ingest the content of input files (of the form **rwyymmdd.00<d>**) containing SMDR records and creates corresponding output files (of the form **rwyymmdd.00<d+1>)** also containing SMDR records.
* For each SMDR record found in the input file a corresponding nearly identical SMDR record shall be created in the output file. Where the input records and the output records differ is that the last 4 digits of each input phone number shall be replaced with a random selection of 4 digits.

*Note: Some phone numbers are ‘known’ in the sense that the source of the call is from a phone that has been specifically installed to support the application. These numbers are not associated with particular individuals and hence privacy is not a concern. Mangling of such numbers is not required nor desired.*

* The mangling shall be consistent throughout an input dataset, even crossing file boundaries. Example: if the number 6049424321 is mangled to 6049421234, then wherever the number 6049424321 is found in the dataset, then it will consistently be replaced with 6049421234 (even if the dataset contains SMDR records for multiple months).

*Notes:*

1. *Phone number mangling is done to ensure phone number privacy. The mangled output files can and will be used during development phases of this and other projects.*
2. *Currently the organization is using a single PBX. An SMDR file with extension***.001** *indicates that its contents are genuine phone numbers.*
3. *Hence, SMDR files with extensions* **.002,. 003,** *etc*.*contain* only *mangled phone numbers.*
4. *It may be useful to think of SMDR files which have an extension* **.00x** *(x > 1), as being sourced from a virtual PBX.*
5. *In principle, it is only necessary to mangle an SMDR .***001** *file once (to a 0.002 file), yet the software shall be able to ‘re-mangle’ .***002** *files to .***003** *files, .***003** *files to .***004** *files, etc. This capability is required in order to allow for off-site testing of the mangling software itself.*

## Test Tool Requirements

A number of related test tools are required to support various stages of testing. The requirements for these tools are discussed next.

### PBX Simulator

The PBX Simulator will primarily be used to test the TCS, however, it will have one other import role: the actual ingestion into the database of all historical SMDR records.

The PBX Simulator works as follows. It accepts the following inputs:

1. An input directory of SMDR files (it will accept files names of the following form: **rwyymmdd.00<d>** (and ignore all other files).
2. An IP address of the platform on which the TCS is running; and
3. The Port number on which the TCS is listening.

If any of the above parameters are found to be missing or invalid (e.g. providing an illegal IP address), then the TXS shall abort.

### Telecom Management System Simulator

The TMS Simulator listens for a flow of messages from the TCS. Its functionality is minimal and is largely provided to server as a data sink for the TCS during testing. Nevertheless, it will parse the incoming flow of messages looking for valid SMDR messages and send them to the system console.

## Database Requirements

The prime focus of the TCS is to capture telephony data and record this data to a database. This database in question is the first such to record operational data on a routine basis and lays the groundwork for further projects. As such, it is important to define a stable, reliable, and recoverable database environment. This section provides the database-specific requirements.

1. The TCS shall parse SMDR messages and record (without filtering) its contents to the SMDR table.
2. Should the database be interrupted or otherwise become unavailable, no data loss shall incur; that is, once database operation is restored, the TCS shall successfully deliver all SMDR messages that have been queued during the period when the database was unavailable.
3. A ‘cron’ type automatic backup procedure shall run daily and deliver a backup dataset to the host platform.
4. The database’s ‘log shipping’ files shall be activated and routinely delivered to the host platform.  
   (the combination of the latest backup set and recent log shipping files provide a means for a perfect database recovery should such capability be required).
5. Finally, a PITR (Point in Time Recovery) database recovery process shall be provided (meaning that the database could be recovered to a specific point in time, that is, not necessarily to include the entire backup set).

## Deployment Requirements

The TCS and sundry is delivered via GitHub. The deployment vehicle is Docker and related tools. This sub-section articulates the deployment requirements:

* The TCS functionality shall be delivered as Docker containers (microservices). This includes both the queuing service, RabbitMQ, and the database, Postgres, as well as the project-specific microservices.
* The consequence is that no external software, other than Docker itself, needs installation.

## Test Requirements

Testing of the TCS is done on two levels:

**Unit Testing**  
Using the Mocha Test Framework, suitable Test Cases will be crafted that will exercise the internal machinations of the TCS. Code coverage is required.

**Test Management**  
A separate Test Management Plan to be composed. It will describe in detail the various Test Cases to be carried successfully in order to ensure that the TCS is painlessly introduced into the operational environment.

This Test Management Plan shall also identify a minimum suite of (priority) Test Cases that need to be successfully executed before subsequent upgrade versions of the TCS can be released into operations.

## Performance Requirements

The minimum performance of the TCS is the following (these performance requirements are not particularly demanding):

1. Able to ingest and deliver to the TMS 1000 SMRD messages per hour for a sustained period of 3 hours.
2. Able to ingest and deliver to the database 1000 SMRD messages per hour for a sustained period of 3 hours.
3. Without message loss, be able to ingest and deliver to the TMS a traffic burst of 200 SMRD messages in 10 seconds.
4. Without message loss, be able to ingest and deliver to database a traffic burst of 200 SMRD messages in 10 seconds.

**Appendix A: Environment Variables**

This section details the various environment variables that need to be set for the TCS to be fully functional. These variables are set in the file **docker-compose.env,** which is located in the project root directory. Note that **docker-compose.env** has been excluded from the GitHub repository for security reasons.

The specific values that are set are only indicative; their values will differ in an operational context.

**BACKUP\_DIRECTORY**The target backup directory on the host platform to which backup sets are to be recorded.

**BACKUP\_SCHEDULE**

The backup schedule defines as a typical cron-like string pattern.

**DOCKER\_MACHINE\_IP**=192.168.99.100

Defines the IP address on which the Docker daemon is running.

**MANGLE\_SOURCE\_DIRECTORY**=../smdr-data/smdr-data-002

The tool Mangle reads files from this directory.

**MANGLE\_TARGET\_DIRECTORY**=../smdr-data/smdr-data-003

The tool Mangle outputs files to this directory.

**PBX\_SOURCE\_DIRECTORY**=../smdr-data/smdr-data-002

The directory from which the PBX-simulator expects to find smdr-files.

**POSTGRES\_PASSWORD**=12345678

The password of the ‘postgres’ database account.

**TCS\_PORT**=3456

The port number on which the TCS is listening for connect attempts.

**TMS\_ACTIVE**=1

As long as the TMS is required, TMS\_ACTIVE should be set to 1. Should the day come when the TMS is no longer needed, then it should be set to 0 (the TCS will need to be rebuilt).

**TMS\_HOST**=192.168.99.100

The host IP address where the TMS is running (note: during development, this variable will often have the same value as DOCKER\_MACHINE\_IP).

**TMS\_PORT**=6543

The port number on which the TMS is listening.