Telephony Requirements Document

Software Requirements Document



R

TAble of  
Contents

2 Introduction 1

2.1 Purpose 1

2.2 Definitions 1

2.3 References 1

3 Overall Description 2

3.1 Product Perspective 2

3.1.1 External Interfaces 2

3.1.2 Internal Interfaces 2

3.1.3 User Interface 2

3.2 Design Constraints 3

3.2.1 Technical 3

3.2.2 Operations 3

3.3 Product Functions 3

3.4 User Characteristics 3

3.5 Constraints, Assumptions and Dependencies 3

4 Specific Requirements 4

4.1 External Interface Requirements 4

4.2 Functional Requirements 4

4.3 Test Tool Requirements 4

4.4 Test Requirements 5

4.5 Performance Requirements 5

4.6 Logical Database RequirementS 5

4.7 Software System Attributes 5

4.7.1 Availability 5

4.7.2 Security 5

4.7.3 Maintainability 5

4.7.4 Portability 5

# Introduction

## Purpose

This document presents the software requirements for a Telephony Capture Service (TSC) that will capture SMDR telephone records from telephone system. The TCS is destined to replace an existing Telecom Management System (TMS), although a transition phase is anticipated in that the new TCS will forward on all SMDR records to the TMS until such time that it is no longer needed.

## Definitions

| Abbreviation | Definition |
| --- | --- |
| SMDR | Station Messaging Detail Record. Its definition can be found in Appendix A. |
| TMS | Telecom Management System |
| TSC | 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. TCS Test Management Plan

# Overall Description

## Product Perspective

### External Interfaces

The TCS has 3 external interfaces:

1. A telephone exchange system, which provides an input stream of messages via TCP/IP. Included in this stream are the SMDR records (other messages other than SMDR records flow on this circuit, although the main TCS function is exclusively interested in SMDR messages).
2. An output interface, also TCP/IP, to a Telecom Management System (TMS).
3. An output interface to a relational database, PostgreSQL.
4. An interface to the Windows Service Control Manager (SCM) which, when configured correctly, will restart the TCS upon reboot, restart it should it about; the SCM also provides an interface to allow a manual shutdown and restarting of the TCS.

### Internal Interfaces

The TCS receives SMDR messages from the telephone exchange and must direct them both to the TMS and PostgreSQL, but either or both of these may not be available and hence it is possible that data may be queued for a considerable period. Further, queued data must be persisted should the TCS itself crash or otherwise needs to be shut down and restarted for some reason. The selected queuing service, [RabbitMQ](https://www.rabbitmq.com/), provides all the of services that are required (once configured currently).

Software Interfaces

* Docker

### User Interface

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

1. A means to both view and edit deployment characteristics. This involves a manual text editor of the so-called Dockerfile that will be provided as part of this project.
2. A means to view the number of items in the two RabbitMQ queues (should one or both of these queues start growing, this is indicative of a problem with the TMS or the database).
3. A means to edit various database configuration parameters and tables as required and according to documentation (which is a project deliverable).

## Design Constraints

### Technical

* Programming language: Version NodejS 6.9.x.
* Database PostgreSQL: Version 10.x.
* RabbitMS: Version: 3.6.x.
* Docker Version: 1.x.

### Operations

ToDo

## Product Functions

## User Characteristics

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

## Constraints, Assumptions and Dependencies

# Specific Requirements

## External Interface Requirements

The TCS shall support an Interface to the Windows Services Control Manager, so that the

## Functional Requirements

In parallel, the TCS shall:

* Receive TCP Segment Data from the telephone exchange and optionally delivers this data unmodified to a persisted queue, TMS\_QUEUE (as supported by RabbitMQ).

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

* The TCS shall receive data from the TMS\_QUEUE and forwards it on to the TMS (i.e. this data once again becomes TCP Segment Data).
* The TCS shall receive TCP Segment Data from the telephone exchange and isolates messages found between the data patterns (in bytes) “00 02 00 00 00 00” and “0a 0d” (carriage return, line feed) and delivers such messages unmodified to two queues:1) DATABASE\_QUEUE and 2) LOG\_QUEUE.
* The TCS shall receive SMDR messages from DATABASE\_QUEUE, parses them into their respective fields (SMDR messages are in a CSV format) and stores them into a database table SMDR\_MESSAGES.
* The TCS shall purge the LOG\_QUEUE, so as to ensure a queue size of between 3000 and 3500 SMRD Messages (note: LOG\_QUEUE is purely provided for contingency purposes and does not contribute to the flow of application data).

## Test Tool Requirements

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

### Create Test SMDR File Generator

* As input this tool ingests the contents of actual SMDR files that have been recorded by the TMS (the files in question are of a specific naming structure: ‘rwyymmdd.001’) and output corresponding test files that can be used later for offsite use (these output files have the naming structure ‘test\_rwhhmmdd.001’).
* For each input file conforming to the required naming convention ‘rwhhmmdd.001’ an output file ‘test\_rwhhmmdd.001’ will be created having identical SMDR records that identical in every sense, except that the last 4 digits of each input phone number have been replaced with a random selection of replace 4 digits.

*The purpose of the above is to ensure the privacy of the number. These ‘test\_rwhhmmdd.001’ files can and will be used during development phases of this and other projects, hence the need for phone number mangling.*

### Telephone Exchange Simulator

The Telephone Exchange Simulator (TXS) will primarily be used to test the TCS, however, it will have one other import role: the actual ingestion into the database of all legacy SMDR records.

The TXS works as follows. It accepts the following inputs:

1. A directory of SMDR files (it will accept files names of the following forms: ‘rwhhmmdd.001’ and ‘test\_rwhhmmdd.001’);
2. An IP address where 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

## 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 Test Cases” that will need to be successfully executed before subsequent upgrade versions of the TCS.

## Performance Requirements

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

1. Able to ingest and deliver to Genesis 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 Genesis a traffic burst of 20 SMRD messages in 1 second.
4. Without message loss, be able to ingest and deliver to Genesis a traffic burst of 20 SMRD messages in 1 second.

## Logical Database Requirements

## Software System Attributes

### Availability

### Security

### Maintainability

### Portability