Seriously Confidential. Dispatch received 2019-09-18 03:04 BST. From: [redacted].

To: Team 1

Cc: [redacted] , Cabinet Office, [redacted]

Due to the recent [redacted] we (SIS) require a clean-room implementation of an SMSC capable of SMPP v3.4 communication.

We have 24 hours to deployment. It is of utmost importance that you implement ONLY what is specified in order to meet this timeline.

We must avoid detection, so all work MUST be completed within normal working hours. Any violation of this condition will result in [redacted] of you and any family members.

Order [redacted] mandates the use of “mob programming” techniques for tasks that fall under Section 13a of [redacted] 2013. The majority of your team’s time must be spent working together on a SINGLE computer. Team members must take up “typist”, “driver” and “advisor” roles, and these must rotate at least every hour. Ask your supervisor if you require more definition of these roles.

You MUST NOT discuss any aspect of your tasks with Team 2, at any time. Remember the [redacted] and [redacted]. Understand the consequences of violating these.

You may choose any technology (platform, programming language, available libraries) for this task, but the code must execute on your supervisor’s laptop at 2019-09-19 11:00 BST. You must write a command-line program that executes in a standard Linux Bash shell.

You MUST avoid using any library code that implements any telecoms-related code, most importantly anything directly related to SMPP, due to [redacted] , obviously.

You MUST implement unit tests* of all aspects of your code, and you must provide when asked an explanation of how your system can support comprehensive acceptance tests.

> ** Unit tests must be capable of executing on our off-grid system, which has no writeable storage or network connectivity.*

For definitive information about the SMPP v3.4 protocol, refer to https://www.openmarket.com/docs/Content/downloads/SMPP-v3-4-Issue1-2.pdf, but for informal understanding purposes we recommend referring to the OpenMarket links given below.

You should expect requests to arrive one at a time. There is no requirement for asynchronous operation.

Complete these tasks in the specified order. Each of these tasks has value, and the release of your ~~loved ones~~ compensation is linked to each task you complete. You are not expected to complete all tasks.

1. Start the program and listen on a TCP port (port number supplied as a command-line parameter).
2. Receive and process a SUBMIT_SM PDU (similar to that described here: https://www.openmarket.com/docs/Content/apis/v4smpp/mt.htm). You may ignore all information in the PDU except Source Address, Destination Address and Short Message. You must print those values to Standard Output so that our operatives may [redacted] under [redacted] within [redacted]. You may assume the short message contains only ASCII characters, and has no TLVs. You may assume the connection is bound: you are NOT required to receive a BIND_TRANSMITTER message first – focus on SUBMIT_SM only. The short message will be encoded using GSM 03.38 encoding, with one byte per character.
3. After processing a SUBMIT_SM, emit a valid SUBMIT_SM_RESP PDU indicating successful receipt of a message (similar to that described at https://www.openmarket.com/docs/Content/apis/v4smpp/mt.htm#submitrespheader). Include only enough information to constitute a valid PDU, indicating successful receipt.
4. After emitting the SUBMIT_SM_RESP, prompt the operative (on the command line), asking whether the message has been delivered, then emit a valid DELIVER_SM PDU (similar to https://www.openmarket.com/docs/Content/apis/v4smpp/deliveryreceipt.htm). If the operative responds that the message was delivered, supply a TLV with tag 0x2153 and value 4 (type: short). If not, supply the same TLV with value 351. The Short Message section of the PDU should begin "id:XXX" where XXX is substituted for a unique message identifier, which matches that provided in the SUBMIT_SM_RESP PDU created in the previous task.
5. Implement optional bind logic (similar to https://www.openmarket.com/docs/Content/apis/v4smpp/bind.htm): if a System ID and Password are provided on the command line, emit no responses until a valid BIND_TRANSMITTER PDU has been received, which contains the same System ID and Password. Once that has been received, emit a valid BIND_TRANSMITTER_RESP PDU indicating success.
6. [redacted]