Remote updating procedures for Mobile point of sale terminals

ALEŠ ZELENIK, ZDENKO MEZGEC
Margento R&D d.o.o.
Gosposvetska cesta 84, 2000 Maribor
SLOVENIA
ales.zelenik@margento.com http://www.margento.com

Abstract: The following article presents an efficient remote updating system that supports software updating through different communication channels. The emphasis is on the robustness of transmission, data size optimizations, quick remote firmware replacing, etc. Besides GPRS and IP based communication, there is also explained a unique way of updating procedure with transmitting the data over speech channel of a mobile phone – DOV. All parts of updating system are thoroughly explained.

Key-Words: - Remote update, speech channel, Data over Voice – DOV, GSM, mobile POS terminals.

1 Introduction

If we wish to ensure normal functioning of systems, we nowadays need suitable and particularly effective approaches to maintenance and updating of software and hardware. This article describes in detail the functioning of algorithms, which have been purposely designed for updating the software of the Margento system. The system is primarily designed for data transfer through speech channels [1][2][6] of different mobile telecommunication networks such as GSM, CDMA and UMTS. For the purposes of communication, the system also uses other communication channels beside the speech channel, the most important and most often used ones being Ethernet and GPRS. The original idea of the system is a patent-protected method of user identification with a mobile phone [4]. Figure 1 shows a simplified scheme of data exchange through a speech channel. The user calls a pre-determined number of the processing centre, puts the mobile phone on the terminal and waits for the sign marking the end of the data transfer, after which the user can put away his/her phone. The processing centre can obtain all the data on requested actions through the exchange, and take further action according to its data. Further information on the operation of the Margento system is available in the articles listed among references.

The present article is divided into four chapters. The first chapter briefly presents the operation of the Margento system. The second chapter presents important limitations that needed to be taken into consideration in the development of the algorithms designed for terminal updates. The third chapter explains the functioning of the update system. The update procedure describes in greater detail the preparation of the data, the transfer of the data to the terminals and the updating methods. Because of a high data-security demand a part of the article also deals with this issue. The last chapter is the conclusion, which sums up the main characteristics of the Margento update system.

Figure 1: Margento system overview [1]

2 Problem Formulation

Several limitations had to be taken into consideration when designing algorithms for updating or maintaining terminal software. The greatest of them were the following demands. The algorithm must securely transfer and confirm all data. The algorithm must be able to update all types of Margento terminals, independently from the current hardware or other software installed on the terminals. The algorithm must enable data transfer irrespectively of the communication capabilities of the terminals. In addition, the size of the data and program memory intended for the update algorithm must be as small as possible.

Because of remote updating, the demands for uninterrupted functioning and a high level of security exist. Consequently, it is necessary to provide additional information blocks, which are to be included with the data. They will detect various errors in the data transfer. Namely, incorrect update data could disable the whole Margento terminal, and physical reprogramming would be necessary, which is an undesirable procedure in several respects.

The next problem is a very low effective speed of the data transfer through the speech channel, which disables transfer of large datasets. Therefore, a compromise must be achieved, which would enable updating smaller parts of the application and not only the whole.

Apart from substituting the data located in separate FLASH sectors, the algorithm must also be able to substitute individual parameters, logotypes, sounds, printouts and fonts, where the new data may be of a different size than the old data. It must be emphasized that this is not a trivial problem, as for data storage, the terminals have a serial FLASH memory installed and not data FLASH memory, which also does not contain the standard file systems. The update algorithm will consequently have to know different intrinsic data formats used for the above-mentioned data types. To substitute a dataset the algorithm will have to be able to read the data of the entire sector onto an external RAM, substitute the desired datasets, adjust the locations of other data and write the combination of the old and the new data back onto the serial FLASH.

In addition to the above-mentioned demands, the algorithm must also be robust and resistant to external influences. Namely, it must be possible to interrupt the process of updating, intentionally or not[1], without this influencing proper operation of the terminal. The updating itself must continue, when certain conditions are again fulfilled.

[1] Shut down, intermediary urgent data transfer, etc.

3 Problem Solution

For the sake of proper functioning of remote updating other largely automated systems needed to be developed apart from terminal algorithms. To achieve proper functioning, changes on several levels were necessary. Figure 2 generally shows the whole procedure of terminal update.

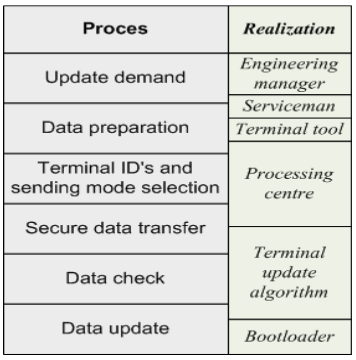


Figure 2: Update system overview

The first step toward a successful update is the serviceman, who prepares the new data intended for the update. The data is then sent to the processing centre, which processes it suitably and sends it to the chosen terminals. Each terminal duly checks and processes the received data and updates its own values with the values prepared by the serviceman.

3.1 Data preparation

An update file containing an intrinsic format must be prepared for proper updating of all or of individual sets of data of an individual terminal. The software Terminal Tool has been developed for this purpose. The program basically supports entering, changing, reading, deleting and marking of the parameters, the program code, logotypes, symbols, printouts and sounds. Figure 3 presents the basic program's window, which shows that the program will generate an update file that will substitute four basic parameters, one logotype and eleven sounds.

The program equips the update file with different security elements, which prevent various errors in updating the terminals. These security elements include both the results of the hash function and signatures as well as the parameters, which must fully correspond to other terminal parameters. These are the parameters of the clock rate, PLL and other hardware parameters[2]. The program Terminal Tool arranges all data that is marked as a new update according to different terminal data formats and simultaneously records the result in the maintenance file, which is saved on the computer's hard drive.

[2] Security elements will be presented in detail in the chapter on updating the terminal.

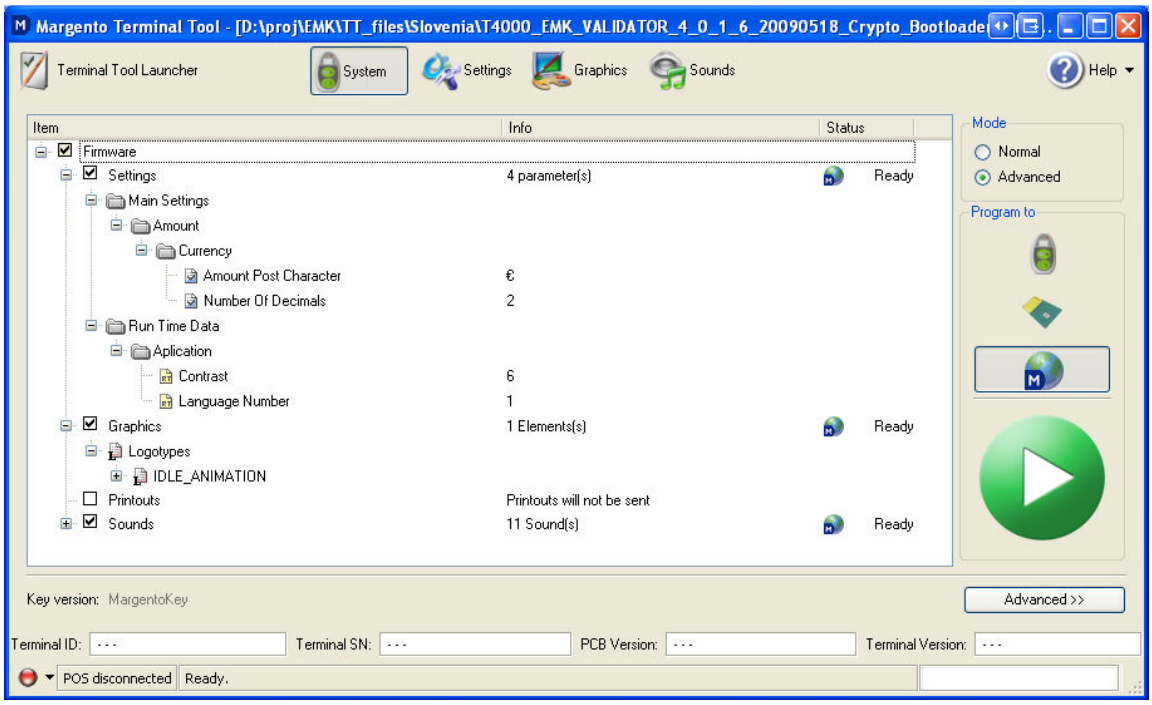


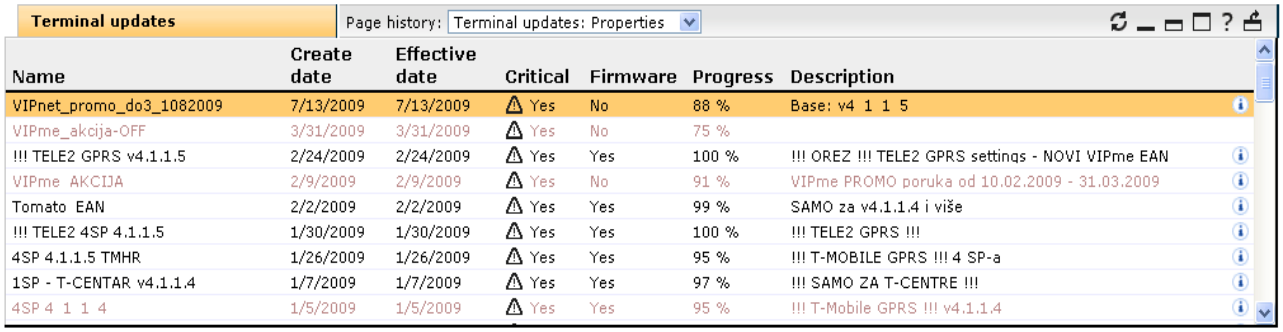
Figure 3: Terminal Tool

All the chosen changes need to be well tested before sending them to normally physically very remote positioned terminals. Although all the changes, especially changes concerning firmware, are usually very well tested, it is still necessary to be careful when creating a maintenance file. Therefore, we always first test the chosen changes included in the update file directly on the terminal in the development department, and only then, we can use the update file in the subsequent steps. High precaution measures are needed because of the fact that we are remotely changing the behaviour of the terminal, i.e. changing the firmware. A mistake could occur and the new firmware could include a bug or even have the functionality corrupted to the point that the new firmware would not be able to support remote update procedures anymore. If this happened, a high number of terminals would become impaired in a very short amount of time, after which they could only be physically reconfigured at every individual location.

3.2 Processing centre

The maintenance file acquired with previous procedure is then, by the serviceman, uploaded to the processing centre. To facilitate the handling and update tracking we have developed a web application called Processing System Administration - PSA. The application operates on several levels. For the purposes of this article, we will only present basic features of the level or the part of the system that is responsible for updating of the terminals.

Before introducing the system that enables remote updating, we have been trying to avoid all software updates as much as it was possible. Namely, every change required a physical presence of the serviceman at the location of each terminal, which resulted in high costs of such changes. This is not the case anymore, due to the introduction of the remote update system. Since then, the number of terminal's updates increased as is shown in figure 4.



Name	Create date	Effective date	Critical	Firmware	Progress	Description
VIPnet_promo_do3_1082009	7/13/2009	7/13/2009	Yes	No	88 %	Base: v4 1 1 5
VIPme_akcija-OFF	3/31/2009	3/31/2009	Yes	No	75 %	
!!! TELE2 GPRS v4.1.1.5	2/24/2009	2/24/2009	Yes	Yes	100 %	!!! OREZ !!! TELE2 GPRS settings - NOVI VIPme EAN
VIPme AKCIJA	2/9/2009	2/9/2009	Yes	No	91 %	VIPme PROMO poruka od 10.02.2009 - 31.03.2009
Tomato EAN	2/2/2009	2/2/2009	Yes	Yes	99 %	SAMO za v4.1.1.4 i više
!!! TELE2 4SP 4.1.1.5	1/30/2009	1/30/2009	Yes	Yes	100 %	!!! TELE2 GPRS !!!
4SP 4.1.1.5 TMHR	1/26/2009	1/26/2009	Yes	Yes	95 %	!!! T-MOBILE GPRS !!! 4 SP-a
1SP - T-CENTAR v4.1.1.4	1/7/2009	1/7/2009	Yes	Yes	97 %	!!! SAMO ZA T-CENTRE !!!
4SP 4 1 1 4	1/5/2009	1/5/2009	Yes	Yes	95 %	!!! T-Mobile GPRS !!! v4.1.1.4

Figure 4: Update orders for a particular area

Figure 4 shows that individual updates include certain additional information intended to facilitate the handling of updates for all the users of the PSA. If we click an individual update, a new window opens, where detail information on the update can be seen, as shown by figure 3.

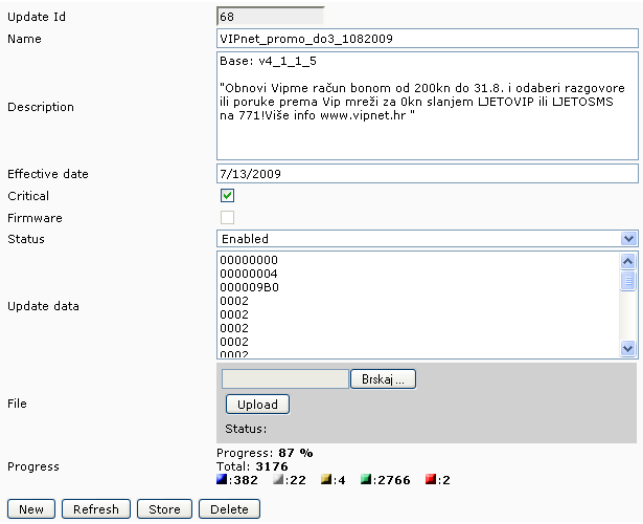


Figure 3: Update file info

The same window as shown in figure 3 is used by the serviceman in the procedure of uploading a new update to the processing centre. Beside the name of the update, the description of the update and the update's effective date are also provided here. The update's effective date enables the software developers to take care of all the necessary software adaptations and upload the update file to the processing centre in advance, although the update should become effective only from a particular date in the future onwards.

To make a further distinction according to the importance of the update and the method of updating the data on the terminal, we have added information on whether the update is of critical importance and whether the update includes firmware. These two pieces of information have a high influence on the update process, that is; "critical" influences the methods for sending the data, while "firmware" determines the steps for selecting proper terminal's update methods. Both procedures will be further described in the following subchapters. After choosing and entering information on the update, the serviceman clicks "store", which records the order for the data in its database and sends the serviceman a confirmation on the receipt of the file and the order.

Individual processing centre communicates with a very large number of terminals, installed at different sales points. In a case when a certain sales point manager requests or orders a change of a particular parameter, this change must only be uploaded to the terminals installed at his premises. This is done in the next step where the serviceman chooses target terminals on which the updates are to be introduced. This is achieved by entering identification numbers of the target terminals in the user interface of the PSA. In case of a greater number of terminals, entering individual identification numbers would be very time consuming, which is why we have implemented filters with which we can mark all the terminals of one administrator at the same time. We can also mark larger groups of terminals, distinguished according to the sales person, distributer, installation location, functionalities and even according to the version of the firmware installed on the terminals.

A great deal of effort has been invested in greater user-friendliness. Thus, the functionality of reviewing the progress of all of the updates for the chosen terminals and their connections has been added to PSA. If we return to figure 3, it also shows the information on the current progress of the update. This concrete example shows that 3176 payment terminals have been chosen for the update, out of which 87% of the terminals have already been successfully updated. The numbers shown next to the icons under the total number of terminals have the following meanings:

- 382 – num. of enabled terminals
- 22 - num. of disabled terminals
- 4 – terminals with update in progress
- 2766 – successfully updated terminals
- 2 – num. of failed updates

Unfortunately, the above-mentioned numbers do not tell us much in terms of error detection or delineate problems with terminal updating. Although we know that in two cases the update has failed, we need exact information, which are these two terminals. This functionality has of course been implemented in sending the data to the terminals, but some wishes have been expressed for the users to have a friendly overview of it. The overview of the terminal status in respect to an individual update has thus been added, as it is shown in figure 4.

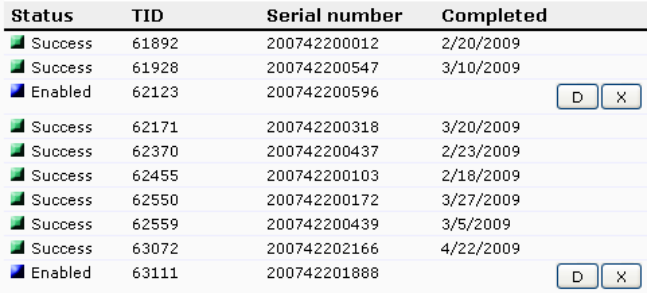
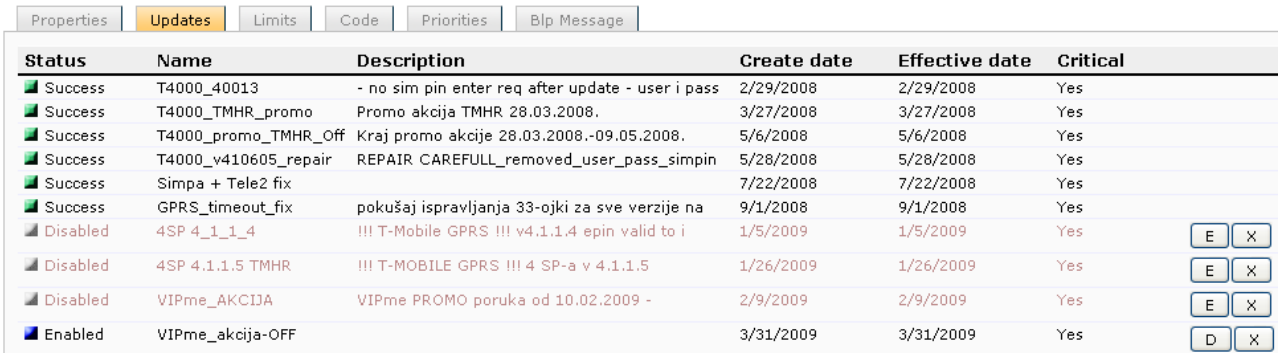


Figure 4: Status of terminals for a particular update



Status	Name	Description	Create date	Effective date	Critical	
Success	T4000_40013	- no sim pin enter req after update - user i pass	2/29/2008	2/29/2008	Yes	
Success	T4000_TMHR_promo	Promo akcija TMHR 28.03.2008.	3/27/2008	3/27/2008	Yes	
Success	T4000_promo_TMHR_Off	Kraj promo akcije 28.03.2008.-09.05.2008.	5/6/2008	5/6/2008	Yes	
Success	T4000_v410605_repair	REPAIR CAREFULL_removed_user_pass_simpin	5/28/2008	5/28/2008	Yes	
Success	Simpa + Tele2 fix		7/22/2008	7/22/2008	Yes	
Success	GPRS_timeout_fix	pokušaj ispravljanja 33-ojki za sve verzije na	9/1/2008	9/1/2008	Yes	
Disabled	4SP 4_1_1_4	!!! T-Mobile GPRS !!! v4.1.1.4 epin valid to i	1/5/2009	1/5/2009	Yes	E X
Disabled	4SP 4.1.1.5 TMHR	!!! T-MOBILE GPRS !!! 4 SP-a v 4.1.1.5	1/26/2009	1/26/2009	Yes	E X
Disabled	VIPme_AKCIJA	VIPme PROMO poruka od 10.02.2009 -	2/9/2009	2/9/2009	Yes	E X
Enabled	VIPme_akcija-OFF		3/31/2009	3/31/2009	Yes	D X

Figure 5: Update list for a particular terminal

Beside the overview of individual update transfer, the possibility of overviewing and changing the updates concerning a specific terminal has been added for the purposes of debugging, as it is shown in figure 5. Here a list of all of the updates and their statuses, for every individual terminal, is available. From this overview of update status and the time for update completion and by taking into consideration the errors occurring during transfer and the number of attempts to send an update to an individual terminal, an algorithm has been added, which can automatically determine the status of the critical update for the update file for every individual terminal. The purpose of the algorithm will be presented in detail in the next subchapter.

3.3 Update data transfer

Communication channels through which the terminal can obtain update-data are divided into two groups:

- slow comm channels (GSM, CDMA),
- fast comm.. channels (GPRS, Ethernet, etc).

For financial reasons most terminals have only the option of a basic method of communication with the processing centre, i.e. communication through a speech channel of the user's mobile phone. The disadvantage of this communication method is that the bit speed of sound modulated data signals is between 300 and 500 b/s. Because of this, these channels are only used for transferring smaller changes: adaptation of configuration or parameters, changing smaller logotypes, etc.

Lately, especially due to limited possibilities when sending update data and for the sake of shortening the time of transaction, faster communication channels are built into most new terminals; the most commonly used being communication via GPRS [3]. With it, the terminal becomes an active communication device, meaning the terminal can connect to the processing centre at any time and exchanges any data.

3.3.1 Update strategies

Most often used terminal's functionalities are various mobile payments, depositing funds on accounts and bonus programs. According to user surveys, using the system for less than five seconds is a good user experience. Transferring additional data during the payment process may prolong this time, which has a negative effect on the user experience.

This problem has been solved by dividing the transfer of update data among several payments or transactions. With it, we can achieve a negligible time prolongation of transactions. The basic data transfer remains the same during payment, but the processing centre additionally sends information on the presence of potential update data. If the centre does not have the relevant order, the transaction is finished in a normal way [2]. If the order exists, the processing centre sends a part of the update data. The processing centre at the same time communicates to the terminal information on whether the update is critical. In this case the terminal will not allow further execution of transactions, and will ask the cashier to make a maintenance call in which the entire update file will be transferred. This call can be made with any telephone.

If there is a large amount of update data, this can consequently mean that updating the terminal will be carried out over a longer time period. The processing centre therefore always conducts an analysis of transactions, so it can foresee such a case, and thus automatically designates the status of a critical update to this update file. The critical status can also be designated by the serviceman who has prepared the data. This usually happens when the update includes essential changes, which should be implemented as soon as possible.

In the presence of faster communication channel also larger updates can be transferred. The main difference in transferring larger updates is the presence of firmware in the update data. The condition for a faster communication channel is the

installation of an active communication device in the terminal, meaning that the presence of a mobile phone is no longer required for the transfer of an update [3]. The data transfer through a faster communication channel is normally 10 to 1000 times faster which considerably shortens the updates transfer time [8]. With faster connections, we have thus waived the option of transferring smaller updates, as transferring the complete update does not affect the user experience. Because of the presence of an active communication device, we were also able to add an automated checking for new update data. Namely with an active connection the terminal can automatically connect with the processing centre without the user noticing this. Checking for new versions can therefore be done during every transaction, as well as in specified time intervals and at every start-up of the terminal.

The presence of a faster communication channel or the necessary use of a slower communication channel must be taken into consideration already when preparing the data. At this step the difference between the above two options is enormous, mainly because of the large difference in transfer speed and possibility of arbitrary connection establishment.

When using a faster communication channel, we mostly use the strategy of sending the entire firmware with all of the parameters, graphics, sounds and logotypes added or we simply include all of the data that is saved on the terminal. This method is, because of its simplicity and simple tracking of the software components installed on the terminals, used with at least 90% of the updates of the terminals enabling communication through a faster communication channel. The other 10% of the updates are those where we basically change all of the data apart form certain data that is specific to the sales point. By keeping that data, we can update a large number of terminals with the same update data, while preserving some degree of the specificity of a certain sales point. These specifics mostly include the logotypes of the sales point, certain minimal special settings etc. These 10% also include the updates of individual components, but we try to avoid these, when having faster communication channels available, as far as possible because of the potential problems, which shall be presented below.

When using a slower communication channel we have, at the very beginning, given up the option of transferring firmware because of much lower transfer data rates [2]. The low transfer data rates also have a high influence on the choice of the update components. If we have mentioned that with faster channels at least in 90 % all data is transferred, in case of a voice channel 75% of the updates include only the transfer of the requested updates, which minimises the amount of update data. With this method of updating, special attention is needed when choosing the elements for the update. Namely, precise control over all the partial updates since the last full update is needed. Only this way we can know which parameters, logotypes, sounds, etc., are installed on a certain terminal. Here information shown in figure 5 is very helpful. Precisely because of this added part, i.e. change tracking, and because of the possibility of potential errors, we usually send all the data needed by the terminal for operation, when faster communication channels are available.

To support all possibilities or update strategies we had to implement additional logics on the processing centre, which handles correct data saving and update realizations on all the corresponding terminals. However, we had to consider the possibility that a request for new software updates might appear before all the terminals have the previous update installed. It should be noted that the lists of updates could vary greatly among terminals as well as the lists of completed updates and the number of updates still waiting for installation. With the queued updates, the difference between partial updates and full updates is larger. If the queued updates include only partial updates, we have no other choice but to consecutively carry out every update. If there is a large number of such partial updates, the processing centre predicts the approximate time or date when all the existent updates should be transferred to the terminal according to the frequency of communication with such terminal [7][8]. If this date is in a very distant future, the processing centre may request a separate service call. This method is specific particularly for transferring updates through the voice channel of GSM network. With faster communication channels, we normally deal with a generally shorter queue of updates on the processing centre. The queue becomes longer with faster channels if the terminal is inactive for a longer period, i.e. when the terminal is turned off. The queue for a faster communication channel mostly includes full updates with a smaller number of interim partial updates. Since every full update overwrites all previous data on the terminal, there is no point in carrying out all updates successively; instead of that, only updates since the last full update are performed. The processing centre detects such cases and with every upload of a new full update deletes all previous updates for an individual terminal from its base. Because the previous upload could have already been almost fully transferred and deletion of this

data would prolong the updating of the terminal, the processing centre may complete the existent update despite the fact there is a new full update queued, and only after completion decides to transfer the new update. With this procedure, we avoid the possibility that a certain terminal would not update because there would always be new full updates present at the processing centre.

3.3.2 Description of actual communication

So far, we have only discussed higher levels of communication between the processing centre and terminals, but have not yet mentioned the actual size of data, the size of packets, time duration of data transfers, etc.

In communication with the processing centre in every packet the terminal separately receives the information on how much data will be received, meaning that this value may change during operation. There are several reasons to support the sending of packets varying in size. The first one is the processing centre workload. The processing centre may, according to the current workload, decide what size of data packets would be most suitable at the time and with what size it would be most efficient in communication with several terminals. The second reason to support the sending of differently sized packets is related particularly to updating through GPRS connection, where usually the GPRS network provider sets an upper limit to the size of the packets, which can be transferred through its network. This maximum size of the packet, which is to be sent through the network, can however be changed by the GPRS service provider, and this can cause problems when the maximum allowed size of the package is reduced. The transfer of all packets that fit the previous size limitation, but do not fit the new limitation anymore because of limitation reduction, could thus be cancelled, which would disable all further communication. The last and probably most trivial reason is the fact that dividing the full update to equal individual packets in most cases would not be possible. The last packet would have to be filled up to the standard size of packets, because of which unnecessary or ballast data would be transferred.

Because of changing the size of the packets, an algorithm capable of identifying and fusing the data had to be introduced also on the terminals. Here several problems occurred. The most obvious problems occurred when changing the size of the packet and simultaneous loss of data or acknowledge for this data. Several reasons may lead to the loss of data. Receiving update data is always performed in the background, as the procedure must not interrupt normal functioning of the terminal. Because it is performed in the background, the administrator does not know that the terminal is being updated and may simply turn off the terminal. Because of this, loss of data or loss of acknowledge may occur. The last two cases may also occur if the network signal is weak, and are even more frequent with transfer through voice channel than with transfer through GPRS. All possible ways of adequate data transfer and loss of data can be summed up in three possibilities.

1. The processing centre sends the data, but the terminal does not receive it or the received data includes errors.
2. The processing centre sends the information, the terminal receives it and records it, but the processing centre does not receive the corresponding acknowledge.
3. The processing centre sends the data, the terminal records it, sends a confirmation to the centre, which receives the confirmation.

Only in the third case, data has been transferred as predicted, while in the first two cases we have to solve several problems. These problems may become even greater if, in the next few moments, the size of packets is also changed. The possible scenarios are shown in figure 6.

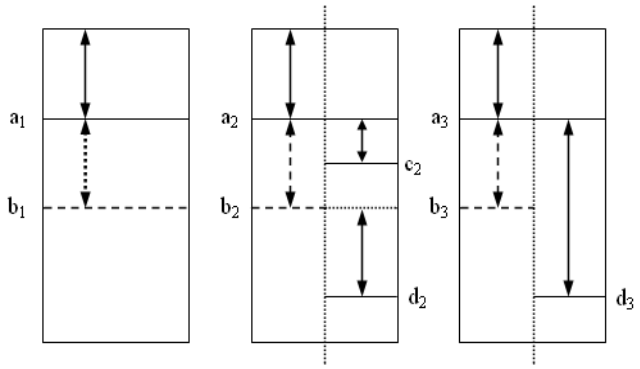


Figure 6: Data loss prevention

The left part shows a scenario where the processing centre sends the data (a_1-b_1) but the terminal does not receive it. Consequently, the processing centre does not receive acknowledge, sends the same data (a_1-b_1) again, and again waits for acknowledge. When sending the data again, the size of the packet may be changed. The other two parts of figure 6 show the second possibility, where the processing centre did not receive the corresponding acknowledge, but the terminal had received the data successfully. The terminal thus holds the data received with previous update packets that is located to point a_2 (a_3), and the update data of the last packet that is saved to point b_2 (b_3). The

processing centre unfortunately does not know this as it has only received acknowledge for the data saved to point a_2 (a_3). Because the processing centre has not received acknowledge for the last data sent, it sends this data again. The terminal detects from the received data that this data has already been received and saved, and thus sends acknowledge to the processing centre while discarding the received data. However, two situations may occur if the processing centre changes the size of the packet when sending the data again:

- the size is reduced (middle part of figure 6),
- the size is increased (right part of figure 6).

With reduced size of the packet, the terminal receives only a smaller amount of already received data (a_2-c_2). This data is confirmed by the terminal and with the acknowledge the information is also sent that actually more data has already been saved and that the processing centre should only send data from there on. The processing centre thus includes the data from b_2 to d_2 in the next packet. In this way, we can avoid sending already received data to the greatest extent. The right part of figure 6 shows the possibility where the size of the packet is increased. The received packet thus includes a combination of previously received data (a_3-b_3) and new data (b_3-d_3). The terminal detects this, and only saves the new data located on the interval between b_3 and d_3

In order the terminal can know exactly when this duplication occurs, and can solve it appropriately, the transfer protocol had to be well thought out. The transfer protocol, beside the actual data, also includes certain additional data on the content of the packet. The update packet includes the following information for sending the update data from the processing centre to the terminal:

- identification code of the packet,
- operation demand,
- data size in packet,
- absolute sent data size,
- actual effective data size,
- CRC.

The identification code of the packet uniformly determines the type of the packet and the data content in it. When terminal receives the code, it knows that the packet includes update data and also knows the structure of the packet. The next information is operation demand. This field is a command determining what should be done with the received data and with the existent update data. Four commands have been used so far within operation demand: “no maintain file”, “clear”, “add” and “update”. By sending the “no maintain file” command, the processing centre reports that no update is available for that terminal at the moment. The system is designed so that individual terminals keep asking the processing centre if there are any updates available, and not vice versa. The command “clear” informs the terminal that the processing centre will start sending a new update file and that all potential data located at the place reserved for the update data must be deleted. This can happen before the previous update has been completed, i.e. in case when a new full update is uploaded to the processing centre during updating of previous updates. Receiving the command “add” the terminal knows that the received data should be added, while the command “update” determines the end of update data and the launch of the update procedure. As already mentioned, the packet always also includes the information on the size of actual data in packet, and the packets are also equipped with the information on the amount of all data sent so far. This information is used by the terminal, especially in situations presented in figure 6, so that the terminal can, even in the cases of lost data, handle the received data correctly. The information about all the data used so far has a double function because of optimization. In combination with the “clear” command, the total number of all the data for the current update is located in this field. The terminal thus knows the whole amount of update data already when receiving the “clear” command, so this information does not have to be added to every packet; and by combining it with the field “absolute sent data size” we have saved a few bits, which would, in case of the “clear” command, always be set to zero. This way we can keep the same structure for all the update packets. In every packet, there is of course the actual data and at the end of every packet, there is a result of a 16-bit CRC.

All the received update packets are confirmed by the terminal using the acknowledge update packet. The latter consists of the following fields:

- identification code of the packet,
- operation request received,
- information on maintenance packet success,
- absolute received data size,
- CRC.

With this packet, the identification code has the same function as in the above-mentioned case. Operation request received confirms the last requested operations the terminal has received. In the information on the maintenance packet, success error codes can be found. There is a large number of these so we shall only mention the most important ones. Beside the message that the received data has been successfully processed, messages on problems can appear, such as: erasing/cleaning of previous

update data failed, data size too large, data corrupted, unknown base address, packet corrupted, update already finished and others. The next information determining the absolute received data size tells the processing centre from which point on the data for the next update packet should be sent, which reduces the amount of duplicated data.

So far, practically all aspects of communication have been described, apart from the actual concrete numbers of the elements themselves. For better understanding, we can mention, that a typical size of the maintenance file holds approximately 1MB of data, which is relatively large for the embedded world, especially if we consider the transfer speeds of used channels. When transferring individual packets we have found that greatest efficiency is achieved when transferring packets with the size 2kB.

At this point one might be interested in how much time it takes for an average update file including firmware to transfer to a terminal. The time of course depends on the connection mode, but from the data analysis at the processing centre, we can conclude that an update is completed on average in 4 to 5 minutes with GPRS connection. With IP connection, this time is even slightly shorter, because of shorter round trip times in the network [7]. Because of different interruptions in the update process, the possibility of poor network etc., all the abovementioned algorithms and procedures had to be introduced. Namely, if the network connection would be interrupted for a moment every minute this would make the entire update procedure impossible, as it would be starting all over again and again all of the time. This way the procedure always continues practically from the last received data. A comparison can be made with the problem of interruption in data transfer from the internet. If unpredicted situations occur there, like computer resetting or disconnection of power supply, the transfer of the entire file has to be started again.

3.4 Security mechanisms

For data transfer, a purposely-designed communication protocol, which includes a number of techniques for preventing intentional and unintentional changing of data (BEC, FEC, interlacing and different cryptographic procedures), is used. Data updating is carried out on the terminal after the last data packet is transferred from the processing centre. Before continuing, the authenticity and correctness of all received data is immediately verified. Further data protection is provided also on other levels. Each update packet has an additional header, checked by the update logics, which ensure proper data processing and verification of the processing centre's logic. All update packets are equipped with a 16-bit CRC to determine errors, and a 16-bit CRC calculated over all data is added.

With updates that are more important[3], the data is equipped with even stronger protection. The first is a 160-bit result of the hash function performed over the whole file. Next is a 288-bit signature, which is a combination of the content of the new firmware and a 128-bit key for signing firmware, which is unique for every individual area. Apart from the above-mentioned protection, there are also other verifications verifying the compliance of different parameter values representing the terminal's hardware. If any error is detected, updating is not performed and the update data is discarded. The update data may also be discarded at the explicit request of the processing centre, usually made by the serviceman.

3.5 Data update

An algorithm is designed for all data updates, which do not include firmware, and is a part of the application level of the terminal. Since the algorithm is a part of the application level, it naturally cannot be used in the process of substituting the firmware. To support the substitution of firmware, the bootloader of the terminal had to adapted, so that at next rebooting it is able to determine whether the serial FLASH includes new update data. Figure 7 presents a block scheme of the software part of the terminal, which shows the main differences between both update procedures [5].

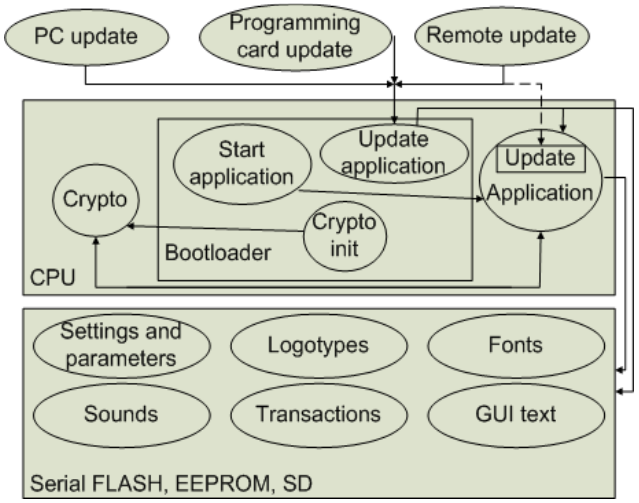


Figure 7: Block scheme of terminal's SW

[3] Particularly with updating the terminal's firmware.

3.5.1 Component update

The update algorithm, which is a part of the application level of the terminal, is in charge of updating individual components. The algorithm must understand the data formats to be able to substitute the data, which is why certain additional information on the structure of the data is included in the update data.

When substituting data we face several problems. Substitution of different parameters is normally not problematic, as only the value of the parameter is changed while the type of the parameter stays the same, meaning that even after the update such a parameter takes up the same amount of space on the serial FLASH or on the EEPROM. Problems occur in substitution of non-standard components, such as logotypes, fonts, printouts or sounds. The possibility that, for example, a changed sound would include exactly the same amount of data as the original sound is very slight. It may occur that the new sound includes less data or that the need for space is greater. The solutions to both problems are very similar.

As we have noted before, the data is stored on the serial FLASH considering a certain internally determined format. The format includes the pointers to individual datasets in the beginning of each sector. When we wish to substitute datasets of different sizes, new data must be properly inserted among the old data. This is done by composing a target sector of data in the main memory – RAM as follows. Data is uploaded from the serial FLASH to RAM. When we reach the point where the data, that is to be substituted is, we instead of the original data, copy the update data intended for substitution. After copying the new dataset, we again begin to record the old data, which follows the substituted data, and memorize the value of the pointer with an updated location of the old data. This value will have to be updated for all such substituted parts in the beginning of the copied sector after completing the operation. When several sets are present, the above-mentioned procedure is repeated. The procedure is completed when all the data from a certain sector is in the RAM together with the updated pointers describing the format and individual elements. At this point, we only need to write the data back to the serial FLASH.

To prevent potential unpredicted behaviour the terminal always automatically reboots after the update is completed. This replacement process is protected so that the algorithm can, in spite of unforeseen external influences already mentioned, continue and automatically complete the interrupted update process.

When substituting a dataset with data that takes up more space, it is possible there would not be enough space to record these data on an individual sector. Terminal Tool therefore includes a protection, which disables the creation of update data file in such cases.

3.5.2 Firmware update

Firmware update is done in a completely different manner than updating individual components. Not only the data but also the basic operation of the terminal shall be updated in this case. The update is therefore done by the bootloader and not by an algorithm of an application level. To successfully store the entire firmware, the terminal must include a larger external serial FLASH memory. A greater demand for space arises from the fact that both the old and the new program code must be saved on the terminal at the same time. The algorithm must first check the basic protection mechanisms on the application level, after which the bootloader continues the update. This is done by rebooting the terminal. The bootloader is activated with every terminal booting. It checks if updates are available or if external devices wish to use it for a certain job. The changes the bootloader must consider may be in the following three locations:

- on the serial FLASH memory,
- on the program card,
- through commands of the serial comm.

In a case of a remote update procedure, the bootloader will find the update data on a specific location on the serial FLASH memory. The bootloader also checks other additional cryptographic protection elements. If all verifications are successful, the bootloader starts erasing the data located in the serial FLASH memory of the DSP, the external serial FLASH memory and on the EEPROM. The left part of figure 8 shows a screen with all the main information on the update process. When all the data is deleted from the terminal, the bootloader starts writing the update data in appropriate places. If unexpected disconnection of power supply occurs while writing data, this data will be deleted at the next start-up and update procedure will start again. All the original update data is saved in a reserved place on the serial FLASH memory until the terminal copies this data to appropriate locations and performs the diagnostics, whereby the final verification of functionality of the received data and firmware is performed. Only after successful final verification, the terminal begins deleting the update data, as it is shown in the right part of figure 8.

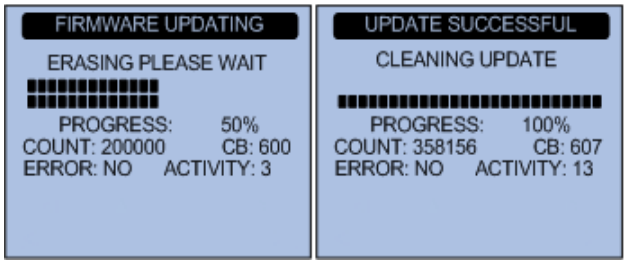


Figure 8: Info on screen

After the data is overwritten, verified, and the update data cleared, the terminal reboots with an updated firmware installed. The final information on successful update is shown in figure 9.

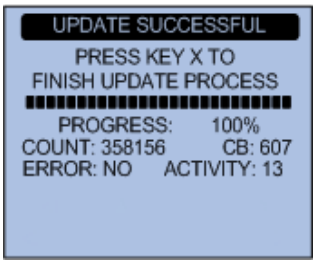


Figure 9: Update finished

The procedure of changing the entire firmware and other data has been designed with the aim of fastest possible completion. It is possible that during the opening hours of sales points the necessity to change some functionalities of the terminal is ascertained. As the processing centre marks such an update as critical with immediate effect, the terminal must become operational again as soon as possible. With well thought out procedure we have thus succeeded in changing the entire firmware of the terminal and other data in only 18 seconds. This is of course an excellent result, as it does not interrupt business operations in any way. Most of this time is consumed by the deletion of sectors of the FLASH memory, where the speed of the deletion, depends on the predefined manufacturer's algorithms. Verification of safety elements, such as CRC, data decryption and verification of keys are performed in a bit over one second, while recording the new data takes less than 5 seconds. The cleaning update procedure has been moved to perform in the background after the terminal has rebooted, which saves a few extra seconds.

To enable good portability of update algorithms among different platforms, they have been developed in a way that they include only a small part that is connected to the structure of the recorded firmware code. All the important information on the structure is already included in the update file. The algorithms are structured so that they dynamically set all the important aspects (CS, setting of the SPI bus, setting of the serial bus, memory organization, data formats, etc.) according to the data in the file. This information gives the bootloader all the necessary data on the terminal, which enables a greater portability of its program code among several platforms.

Finally, we should mention figure 10, which shows from the terminal's perspective, the list of changes necessary for successful completion of the update procedure. Most of these changes have been discussed in the article, so the figure 10 should only serve to help imagine the final image of the necessary changes and to asses the complexity of the system.

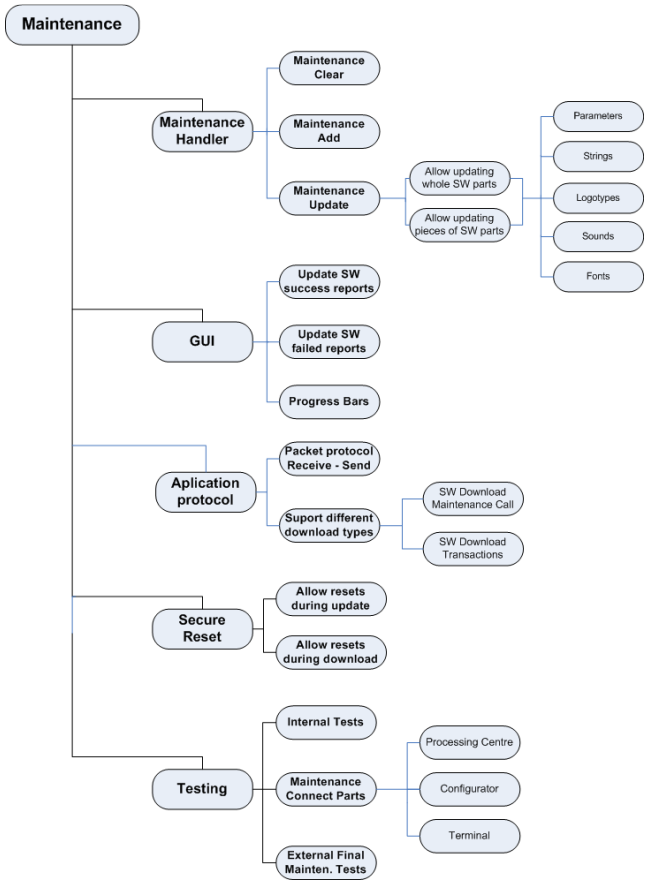


Figure 10: Overview

4 Conclusion

The implementation of a remote data update system has resulted in great advantages in terms of simpler and more professional maintenance of the Margento system. Because of the ease of its use, fast responsiveness and other advantages, the algorithm immediately became a valuable part of the Margento system and enhanced its applicability and adaptability. The concept of the three parts of the algorithm, where the first takes care of the data transfer, the second verifies the data security and the third updates the data, has proven to be the best solution.

By using the algorithm, we now have the possibility of remotely and without any physical human intervention, changing the data on the terminal. In extreme cases, it is also possible to completely change the behaviour of the terminal. Thus, we can for example, change the functionality of the terminal, which has been used, apart from payments, for printing prepaid accounts filling cards into a terminal, that enables automatic filling of prepaid accounts and also storing bonus loyalty points. The change of functionality can of course be even more radical, as the only limitation is now in the terminal's hardware.

The presented update algorithm has also resulted in an additional advantage, i.e. a simplified testing of new program modules. The basic testing of new program modules is still done in the Development Department, but at the same time, we can perform controlled tests on the real system on a smaller number of test terminals. The results of such testing are collected at the processing centre, where the developers can evaluate them and compare them to the functioning of older terminal versions. This way advantages and disadvantages may be determined through a comparison of functioning under real conditions. The absence of wide testing under controlled conditions of development laboratories is also very cost-effective, and there is no uncertainty over the actual behaviour of the terminals in reality when introducing changes to all terminals, since the answer to this has already been revealed.

References:

[1] http://www.margento.com/.
[2] Z. Mezgec, M. Pec, R. Svečko, A. Chowdhury, Data Transmission over the Speech Channel of GSM System, ERK'05, Int. Electrotechnical and Computer Science Conf. (in Slovenian with English abstract).
[3] F. Horvat, D. Slavinec, Z. Mezgec, A. Chowdhury, Upgrade of the basic M-Pay communication, ERK'07, Int. Electrotechnical and Computer Science Conf. (in Slovenian with English abstract).
[4] Z. Mezgec, A. Medved, A. Chowdhury, R. Svečko, Mobile payments – development of a new terminal, MIDEM'08.
[5] H. Hu, H. Hu, Research on Protocol-Level Behavioral Substitutability of Software Components in Component-based Software System, WSEAS transactions on computers.
[6] M. Rashidi, A. Sayadiyan, A New Approach for Digital Data Transmission over GSM Voice Channel, WSEAS'08, CISST.
[7] C.M. Sarraf, L. El-Khazan, Measuring QoS for GPRS Mobile Networks. WSEAS'05, Int. Conf. on Telecommunications and Informatics.
[8] J. Pylarinos, S. Louvros, K. Ioannou, A. Garmpis and S. Kotsopoulos, Traffic Analysis in GSM/GPRS Networks using Voice Pre-Emption Priority, WSEAS'05, Int. Conf. on Mathematical methods and computational techniques in electrical engineering.