Telehealth Monitor
RTX337x

Technical Reference Manual

Confidential Information

1 Indications for use

Indication for Use Statement

Device name: RTX3370
Indications for Use:
The RTX337x is for use in non-clinical settings (such as the home), as an accessory device that is intended to be a communication tool to enable healthcare providers to receive historical patient information. It is intended to be used in combination with a variety of external devices. The RTX337x serves as the remote communication link between compatible external devices, and the compatible healthcare facility at another location. The healthcare facility could be at a disease management center or with the healthcare/wellness provider or other out of hospital caregivers.
The purpose is to collect and transmit selected medical information (such as weight, blood pressure, blood glucose) over a normal residential telephone line. The RTX337x Monitor does not measure, interpret or make any decisions on the vital data that it conveys.

Device name: RTX3371
Indications for Use:
The RTX3371 is for use in non-clinical settings (such as the home), as an accessory device that is intended to be a communication tool to enable healthcare providers to receive historical patient information. It is intended to be used in combination with a variety of external devices. The RTX3371 serves as the remote communication link between compatible external devices, and the compatible healthcare facility at another location. The healthcare facility could be at a disease management centre or with the healthcare/wellness provider or other out of hospital caregivers.
The purpose is to collect and transmit selected medical information (such as weight, blood pressure, blood glucose) using standard wireless technologies. The RTX3371 does not measure, interpret or make any decisions on the vital data that it conveys.

Device name: RTX337x
Contraindications, precautions and warnings:

- This device is not intended for emergency calls, and may not be used for transmission or indication of any real-time alarms or time-critical data.
- All patient medical diagnosis and treatment are to be performed under the supervision and oversight of an appropriate healthcare professional.
- This device is not for use in systems, which substitute for medical care.
- This device is not intended for patients requiring direct medical supervision or emergency intervention.

Table of Contents

1 INDICATIONS FOR USE 2

2 GENERAL INFORMATION 7

3 SAFETY INFORMATION 8

3.1 PRECAUTIONS AND WARNINGS 8
3.2 IMPORTANT INFORMATION 8
3.3 NOTE! 9

4 INTRODUCTION 10

5 SYSTEM DESCRIPTION 11

5.1 GENERAL DESCRIPTION 11
5.2 COMMUNICATION WITH THE RTX337X 14
5.3 SUPERVISION AND TIMING 18

6 GETTING STARTED 20

6.1 PARTS IDENTIFICATION 20
6.2 INSTALLATION STEP BY STEP 21
6.3 CONNECTING THE PSTN PHONE LINE (RTX3370 ONLY) 22
6.4 INSERTING SIM CARD (RTX3371 ONLY) 22

7 CONFIGURING THE RTX337X 24

8 PROGRAMMING AT+P COMMANDS 27

8.1 AUTHORIZING SYSTEM LOGIN (USERNAME / PASSWORD) 27

9 AT+P COMMAND SET 31

9.1 COMMANDS FOR RTX337X CONFIGURATION 34
9.2 COMMANDS FOR RTX337X COMMUNICATION 59
9.3 SECURITY COMMANDS 80
9.4 SERVICE COMMANDS 83
9.5 TESTING COMMANDS 95

10 JAVASCRIPTS AND MMI 98

10.1 MAKING SOUND FILES 98

10.2 MAKING IMAGE FILES ... 99
10.3 UPLOADING JAVASCRIPTS, SOUND FILES, IMAGES AND FONTS ... 99
10.4 USING TEXTS, IMAGES AND BUTTONS ... 100
10.5 TRUETYPE FONTS ... 106
10.6 SCRIPT MEMORY ... 108
10.7 USER RESPONSE MESSAGE FROM SCRIPT ... 108
10.8 CUSTOMIZED JAVASCRIPT METHODS ... 109
10.9 JAVASCRIPT SCHEDULING ... 141
10.10 INITIAL INITIALIZATION OF RTX337X ... 144
10.11 JAVASCRIPT EXAMPLES ... 148

11 WEB SERVER INTEGRATION ... 171

11.1 GETTING STARTED ... 171
11.2 STEP BY STEP CONFIGURATION ... 171
11.3 SAMPLE APPLICATION PROJECT ... 176
11.4 GETTING DATA ON THE WEB SERVER ... 177
11.5 WRITING THE APPLICATION ... 188
11.6 GZIP ... 189
11.7 SSL ... 190
11.8 CONFIGURING THE RTX337X FOR SERVER CONNECTION ... 196

12 XML ... 197

12.1 WEB SERVICE DESCRIPTION LANGUAGE ... 197
12.2 XML PART ... 198
12.3 BINARY PART ... 216
12.4 SERVER COMMUNICATION SEQUENCE ... 216

13 INSTALLATION OF EXTERNAL DEVICES ... 218

13.1 EXTERNAL DEVICE DRIVER ENABLING ... 219
13.2 A&D BLUETOOTH WEIGHT SCALE ... 220
13.3 IEM BLUETOOTH WEIGHT SCALE ... 226
13.4 A&D SERIAL WEIGHT SCALE ... 231
13.5 THT BLUETOOTH WEIGHT SCALE ... 235
13.6 THT SERIAL WEIGHT SCALE ... 239
13.7 A&D BLUETOOTH BLOOD PRESSURE MONITOR ... 243
13.8 IEM BLUETOOTH BLOOD PRESSURE MONITOR ... 248
13.9 A&D SERIAL BLOOD PRESSURE MONITOR ... 253
13.10 ROCHE BLOOD GLUCOSE METERS AND COAGULATION MONITOR ... 257
13.11 LIFESCAN ONETOUCH ULTRA BLOOD GLUCOSE METER ... 265
13.12 LIFESCAN ONETOUCH ULTRA 2 BLOOD GLUCOSE METER ... 270
13.13 LIFESCAN "ONETOUCH ULTRAEASY" / "ONETOUCH ULTRAMINI" BLOOD GLUCOSE METER ... 276
13.14 LIFESCAN "ONETOUCH VITA" BLOOD GLUCOSE METER ... 281
13.15 LIFESCAN ONETOUCH ULTRASMART BLOOD GLUCOSE METER ... 286
13.16 ABBOT/THERASENSE BLOOD GLUCOSE METERS ... 298

13.17 BAYER SERIAL BLOOD GLUCOSE METERS 303
13.18 VIASYS AM1 BLUETOOTH ASTHMA MONITOR 309
13.19 MIR SPIROTEL SPIROMETER 315
13.20 NONIN IPOD SERIAL OXIMETER 324
13.21 NONIN 4100BT OXIMETER 328
13.22 NONIN 9560 ONYXII OXIMETER 332
13.23 MINDRAY PM50 OXIMETER 337
13.24 MINDRAY PM60 OXIMETER 341
13.25 CORSCIENCE BT 3/6 AND BT 12 ECG RECORDERS 345
13.26 VIRTUAL DEVICE 351
13.27 A&D UC-355PBT-CI BLUETOOTH SCALE 353
13.28 A&D UC-351PBT-CI BLUETOOTH SCALE 356
13.29 A&D UC-767PBT-CI BLUETOOTH BPM 359

14 LABELLING AND MARKINGS 362

RTX3371 TYPE LABEL: 364

15 UNIQUE IDENTIFICATION OF YOUR RTX337X 365

16 MAINTAINCE 366

16.1 CLEANING 366
16.2 REPAIR 366
16.3 REFURBISHMENT 367
16.4 FIRMWARE UPDATE 367

17 RTX337X TECHNICAL SPECIFICATIONS 369

17.1 RTX3370 369
17.2 RTX3371 370
17.3 ELECTROMAGNETIC EMISSION 371
17.4 ELECTROMAGNETIC IMMUNITY 372

18 PHONE LINE REQUIREMENTS 375

18.1 REGULATIONS AND COUNTRY SETTINGS 375

19 TROUBLE SHOOTING LIST 377

19.1 LOGGING 377
19.2 CHECK THE BASICS 377
19.3 INSTALLATION 378
19.4 BASIC USE 378
19.5 SERVICE INTERFACE 378
19.6 PSTN / ISP CONNECTION 379
19.7 SERVER CONNECTION 380

19.8 CONFIGURATION 380
19.9 JAVASCRIPTS AND MMI 381
19.10 USE WITH EXTERNAL DEVICES 382

2 General information

This document and the information contained herein, is property of Tunstall Healthcare A/S, Denmark, and should be regarded as confidential. Unauthorized copying is not allowed. The information in this document is believed to be correct at the time of writing and is subject to change without notice.

The purpose of this document is to provide guidance and instructions to operators of the RTX337x on how to write JavaScripts and configure the RTX337x for specific use, and how to handle the RTX337x in daily use. The document also describes the maintenance process.

The document also provides guidance to technical users of the RTX337x on how to configure a Web server and build an application for data collection. Chapter 11, Web Server Integration, describes 'Step By Step' how a Web server, based on Microsoft Windows 2003 Server, can be setup.

The document does not contain information on how the collected data should be processed, presented, stored and how the patient data should be treated.
The user therefore needs to implement these services into their own healthcare services.

Tunstall Healthcare renounces any obligation with regard to installation or usage of third party software.

3 Safety information

The following general safety precautions must be observed during all phases of operation and service of this product. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. Tunstall Healthcare A/S assumes no liability for the customer's failure to comply with these requirements.

3.1 Precautions and warnings

This device is not intended for emergency calls, and may not be used for transmission or indication of any real-time alarms or time-critical data.
Clinical judgement and experience are required to check and interpret the measurements collected and transmitted. The device is not for use in systems, which substitute for medical care.
The device is not intended for patients requiring direct medical supervision or emergency intervention. Also the follwing precautions must be followed:

- *DO NOT* operate the product in an explosive atmosphere or in the presence of flammable gasses or fumes.
- *DO NOT* perform procedures involving cover removal unless you are qualified to do so. Operating personnel must not remove equipment cover. Procedures involving the removal of cover are for use by Tunstall Healthcare A/S authorized service-trained personnel only.

3.2 Important information

- Use only the original power supply adapter Friwo FW7333M/06 or GlobTek WR9QB1000CCP-N-MED that comes with the RTX337x.
- The RTX337x contains a small lithium battery. Do not replace.
- It is important that the cables be carefully arranged and secured to avoid unintended disconnection, or the possibility of tripping or blocking the path of a wheelchair or other moving objects.
- Be careful not to use a power socket that may be switched off unintendedly.
- Do not install or service the device, power adapter or cables during an electrical storm.
- Do not place the RTX337x in extreme climatic conditions (excessive heat or humidity, in direct sunlight or where it may be exposed to large amounts of dust).
- Do not place beverages near the RTX337x as spills may easily result in serious damage.
- Protect the device and cables from damage.
- Keep pets and small children away from the area where the RTX337x and cables are located.
- Equipment not suitable for use in the presence of a flammable anaesthetic mixture with air or with oxygen or nitrous oxide.

The RTX337x and some of the external devices contain radio transmitters and receivers based on Bluetooth Wireless Technology. For optimal performance it is therefore recommended that:

- The RTX337x is placed at a central location in the house.
- Do not to cover the RTX337x behind metal shielding objects like heating panels, refrigerators etc.
- The RTX337x must be placed where a phone plug is within range. Alternatively an extra cable can be used to reach the phone wall plug.

3.3 Note!

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna of the other equipment.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from the one used for the receiver.

If the problem still occurs, consult the manufacturer or an experienced radio/TV technician for help.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

- This device may not cause harmful interference, and
- This device must accept any interference received, including interference that may cause undesired operation.

Modifications not expressly approved by this company could void the user's authority to operate the equipment.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna of the other equipment.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from the one used for the receiver.
- Consult the manufacturer or an experienced radio/TV technician for help.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation. Modifications not expressly approved by TunstallHealthcare could void the user's authorityto operate the equipment.

4 Introduction

This technical reference manual contains the information that is needed for general-purpose use of the RTX337x, and also detailed description of how to prepare and set up the RTX337x for general-purpose use and for maintenance and trouble shooting.

The manual provides programming guidance to the operator of the RTX337x. This information is supplied only as guidance to ease customer's program development.

The manual also shows you how to install and operate the RTX337x from the Windows® HyperTerminal service interface from a PC or using remote command applications, via the SOAP communication protocol.

The following text conventions are used in this manual:

RTX337x	The RTX337x Telehealth monitor.
MMI	Man-Machine Interface.
Home monitoring solution	Means a system for Remote Monitoring of Vital Signs consisting of a RTX337x and one or more external devices.
Data Host Server/ Host Server	Means a HTTP server with continuous Internet access, installed at the Health Service Provider, collecting, handling and presenting the data sent from each in-field RTX337x. Also configuration changes and maintenance is done by the Host Server.
ISP	Internet Service Provider. The suppliers of Internet access, such as AOL or WorldCom.
Internet	Means the global Internet.
PSTN	Public Switched Telephone Network
Operator	Means a person granted the highest level of system authorisation (username+password) in order to configure and make changes to the system configuration and set-up.
User	Means a person granted the lowest level of system authorisation (username+password) in order to configure and make changes to the system configuration and set-up.
Installer	Means the person who installs the home monitoring solution at the patient.
Nurse	Means the home health nurse, visiting the patient regularly.
Patient	Means the patient actually being monitored by the home monitoring solution.
Health Service Provider	Means the entity that runs the HTTP server which collects data sent from each in-field RTX337x.
Gateway	Means the RTX337x
JavaScript	Script containing JavaScript code.
External device	Means a device that operates with the RTX337x such as weight scales, blood pressure meters etc.

5 System description

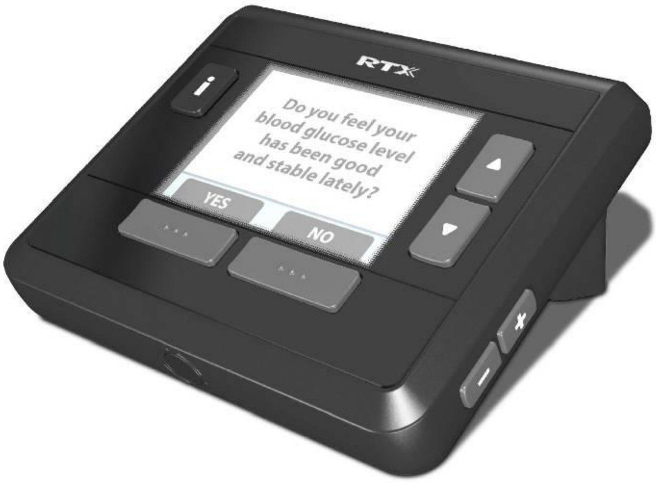

RTX
Do you feel your blood glucose level has been good and stable lately?
YES
NO

Figure 1. The RTX337x Telehealth Monitor.

The RTX337x is for use by patients at home. It is intended that the RTX337x is used in combination with a variety of external devices upon the prescription of a licensed physician or other authorized healthcare provider. The RTX337x is used for wireless or cabled reception of measurements made by on-site external devices such as a blood pressure monitor, a personal weight scale or a blood glucose meter. The RTX337x serves as the remote communication link between compatible external devices and the compatible healthcare facility at another location.
Data is sent over the Internet using encrypted and secured data protocols. Authorized personal at the Service provider's side, like nurses, can access these data. The transmission of data from the RTX337x to a remote facility can be configured to work in fully automatic mode, and requires no user interaction.
Besides being able to transfer measured data from external devices, the RTX337x can also be configured to pose questions for the patient and transmit the collected responses.

The system is communicating through different protocols to ensure a secure data transmission and also to create an easy interface to the existing local area network. Healthcare organizations, hospitals and homecare providers collect data in databases and maintain the system.

5.1 General description

The RTX337x is a device for remote patient monitoring. The RTX337x device is able to:

- Receive questionnaires, configuration etc. from a Host Server via PSTN and Internet in accordance with the XML protocol, see chapter 12.
- Display dynamic patient questionnaires and collect replies from patients. Questions are displayed as text on the display, and verbalised using a built in speaker. The patient can reply to questionnaires by pressing buttons.
- Receive biomedical data from various external devices – either wireless via Bluetooth or IR or via RS232 cable in accordance with Installation of External Devices, chapter 13.
- Store and forward patient data and answers to Host Server via PSTN and Internet as described in chapter 11, Web Server Integration.

The RTX337x is designed to be very flexible and to support a wide range of possible use scenarios. That means, that the RTX337x basic firmware provides the possibilities, whereas the actual functionality of the RTX337x in a specific patient environment is determined by the way the operator decides to utilize the possibilities provided by the RTX337x.
The operator must prepare the RTX337x for a certain way of operation by means of JavaScripts, texts, images, sound files and configuration to determine exactly what to happen on exactly which events in the RTX337x.

5.1.1 JavaScripts and configuration

The RTX337x allows the user to interact via display, sound and buttons. It is possible to present information to the patient via the display and via sound. It is possible to ask the patient questions via display and sound and to react on patient answers (button presses).
The RTX337x user interface is implemented by means of JavaScript. The JavaScript code can react on specific RTX337x events that are generated as responses to user interaction. Specific events on the RTX337x (e.g. button presses, external device connections, received measurements, time outs, etc.) are implemented as configurable events.
For these events it is possible to assign a JavaScript that will be activated when the event occurs. The priority is configurable and if a JavaScript with a lower priority is active at the time of a new event, the active JavaScript will be notified that a higher priority JavaScript would like to run. Then the active JavaScript is responsible of finalizing its activities and leave the arena for the new higher priority JavaScript (for more information, see section 10.9.1).
All display text and images are configurable as part of a JavaScript (e.g. questions, statements, information, reminders, images etc.) and all relevant RTX337x parameters can be shown within dynamic display text lines and images.
The display is divided into different areas for text and images, see section 10.4.1, text and image positions, for more details.
Events in the JavaScripts can be used as condition for routing in the JavaScript. Event responses in a JavaScript can be recorded (e.g. specific button pressed, specific value selected, etc.) and returned to the decided destination (Host server or specific RTX337x parameter).
Audio can be played if a text line refers to an audio file (e.g. when the text is shown on the display).

By programming the JavaScripts and configuring the RTX337x, the operator determines e.g.:

- What happens during initialization in the patient’s home the first time the RTX337x is powered on.
- Which information and reminders are sent to the patient - and when.
- What happens based on patient data received from external devices - and when.
- Which questions are raised to the patient (and when) - and among this:
    - Which texts are shown in the display.
    - Which phrases are verbalised/voiced by the speaker.
    - Which images are shown in the display.
- What happens based on button-press answers received from the patients - and when.
- What happens at a certain fixed time, time-out or time interval.
- What happens if something else is not happening - e.g. inactivity for a certain period.
- What initiates a connection from the RTX337x to the server - and when.

Programming JavaScripts and defining the Man Machine Interface (MMI) is explained in details in chapter 10 in this manual.

In order to use the RTX337x it is necessary to configure the RTX337x. How to do this, and how to program the RTX337x through AT commands, is described in chapter 0 to 9. Download of software to the RTX337x can be done via a PC and the service interface or remotely from the server via the Internet and PSTN. The description of how to do this is explained in details in section 9.1.17 and section 16.4.

The RTX337x can be in three different modes, depending on the application:

- **Normal mode**
- **Patient mode**
- **Factory mode (default mode)**

Each mode has its own default JavaScript. For more information on RTX337x modes, see section 10.10.1.

5.1.2 Host Server

The concept Host Server covers the entire backend of the system. Generation and management of questionnaires, events, reminders etc. as well as the configuration of the RTX337x and the presentation of data is taking place on the Host Server side.
In order to obtain communication between the RTX337x and the Host Server the following must considered:

- All RTX337x interaction with the user has to be configured. It is not real time interaction. Multiple preconfigured JavaScripts will be activated on desired events. The recorded actions will, when the patient has run through a JavaScript, be sent to the Host server along with a JavaScript identification.
- Identification and presentation of JavaScript answers and external device measurements for the medical trained staff.
- Management of the assignment of JavaScript to RTX337x events.
- Deciding the behavior of when JavaScripts has to be skipped, queued or interrupted (maybe with priority, see more in section 10.9.1).
- Generation of text and images to display and audio files to be played.
- Management of the RTX337x database with JavaScript files, text-line files, audio files and image files, to ensure that files needed by the configuration is present.
- Configuration and connection setup of External Devices.
- Configuration of Timer events.
- Configuration of ISP connection.
- Configuration of Host Server connection.
- Configuration of Host Server connection retry scheme.
- Configuration of Patient data.
- Real time clock synchronizing.

5.2 Communication with the RTX337x

Communication can take place on both the 'Server' side and the 'Patient' side of the RTX337x, see figure 2. Data, such as blood pressure measurements, weight measurements etc., is directly transferred from the external devices via the RTX337x to the Host Server.

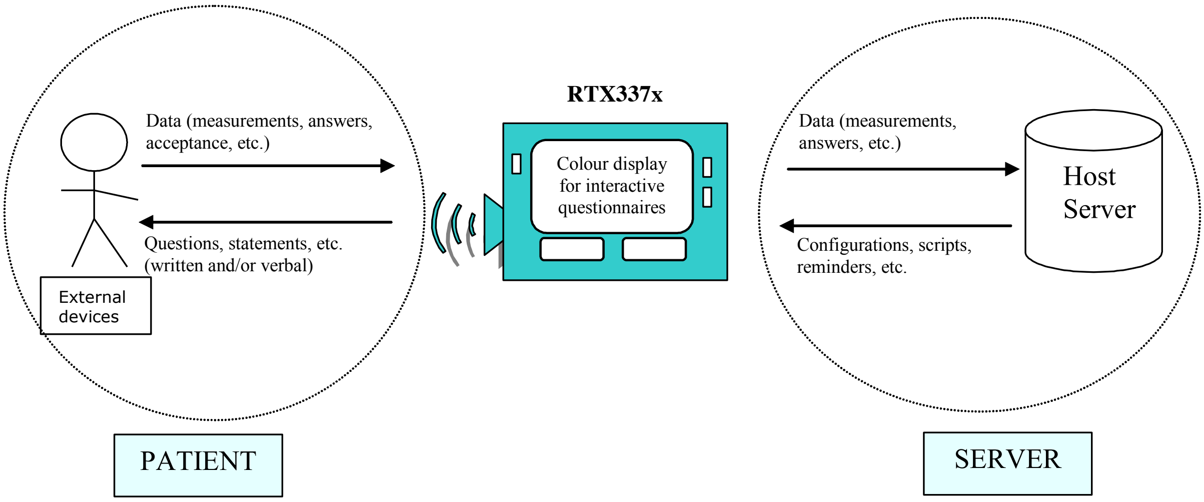

RTX337x
Data (measurements, answers, acceptance, etc.)
Questions, statements, etc. (written and/or verbal)
External devices
Colour display for interactive questionnaires
Data (measurements, answers, etc.)
Configurations, scripts, reminders, etc.
Host Server
PATIENT
SERVER

Figure 2. There is communication on both 'sides' of the RTX337x. The 'Server' side is illustrated on the right and the 'Patient' side is illustrated on the left.

5.2.1 Communication on the Server side

Transmission of data from the RTX337x is automatically initiated by the RTX337x, by establishing a connection to the Internet over PSTN. The "road" for data transmission between the Server and the RTX337x is illustrated on figure 3.

By using the SOAP protocol over a TCP/IP transport protocol secured by means of SSL (Secure Socket Layer), the data is transferred directly to the Web server in a XML format (refer to information in the chapter 12, XML). For information on how to configure and integrate a Host Server see chapter 11, Web Server Integration.

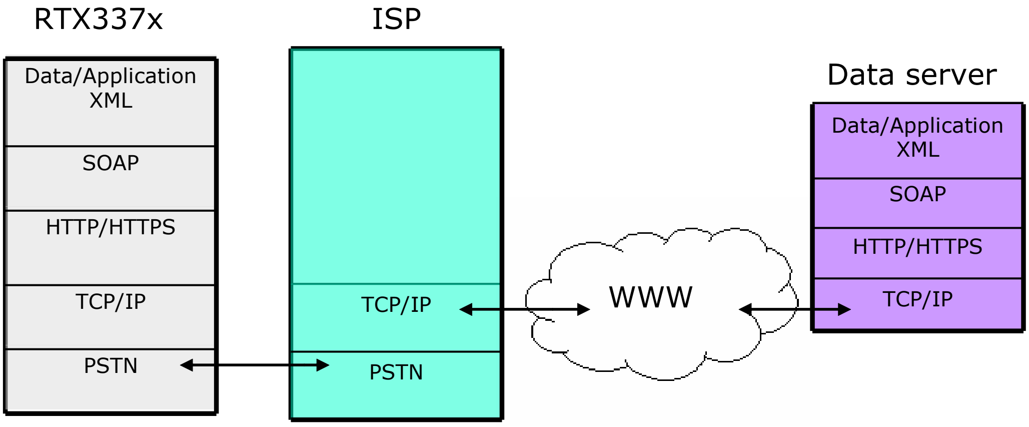


Figure 3. Illustration of data transfer between the Server and the RTX337x.

The Server communication is a two-way communication. Data can be sent from the RTX337x to the Server and data can be sent from the Server to the RTX337x. For any communication to occur the operator must have configured the Server side for communication, see chapter 11, Web Server Integration.

Communication from RTX337x to Host Server

If configured for automatic dialing (see section 9.1.8, AT+pAUTODIAL), the RTX337x will make a server connection if:

- The system has just been powered up (to announce its presence).
- Data from external devices has been received.
- JavaScripts are configured to call server, e.g. 'User responses' to questionnaires.
- A Supervision timeout has occurred.
- An error has occurred.

If *not* configured for automatic dialing (see section 9.1.8, AT+pAUTODIAL) the RTX337x will make a server connection if:

- JavaScripts are configured to call server, e.g. 'User responses' to questionnaires.
- An error has occurred.

System power up

When the system has been setup, the RTX337x will make a server connection to inform the Host Server that the specific RTX337x is operable.

Data from external devices

When the RTX337x has received data from an external device a server connection will be established and the data will be delivered. Depending on the configuration it can be immediate or time dependent. If the connection is not successfully established, the RTX337x will automatically try again. If, for some reason, it is not possible to establish a server connection, the data will be saved in the RTX337x and be delivered at next server connection.

JavaScripts

JavaScripts can be configured to call the server in different scenarios. For example when questions have been answered by the Patient, a JavaScript can be set up to provoke a server call, and the answers or selections can be transmitted. If the answers are linked to a specific measurement from an external device, this will emerge from the data.

Supervision

When a Supervision timeout occurs, the RTX337x will make a server call.

Errors

If an error occurs, the RTX337x will try to make a server connection. The errors that will provoke a server call are listed in section 9.4.17.

Communication from Host Server to RTX337x

The Host Server cannot initiate communication with the RTX337x. The RTX337x will always initiate the communication, though the RTX337x can be configured to initiate a call to the server at a specific time or on other events. Once connection is established the Host Server can communicate to the RTX337x. Communication from the Host Server to the RTX337x can be made in order to:

- Make changes in the configuration
- Update the RTX337x with new JavaScripts (new questionnaires)
- Send reminders to the Patient (e.g. remind a patient to make measurements)

Make changes in the configuration:
From the Host Server it is possible to make changes in the configuration of the RTX337x by sending AT-commands.

Update the RTX337x with new JavaScripts:
It is also possible to download new JavaScripts from the Host Server to the RTX337x. This makes it possible to, for example, change or add new questions for a specific Patient.

Send reminders to the Patient:
It is possible to configure the RTX337x to send time-dependent reminders from the Host Server to the RTX337x (see section 13.18, Virtual device). The Patient can be reminded to make measurements, take medication or to attend doctor's appointments. It is also possible to supervise that an external device occasionally sends data to the RTX337x (see section 9.1.13, AT+pDVS), or if server connections are made from a specific RTX337x to the Host Server (see section 13.18, Virtual device).

5.2.2 Communication on the Patient side

The communication on the 'Patient' side is also a two-way communication. The RTX337x can communicate to the Patient via written and/or verbal questions and statements. The Patient can communicate to the RTX337x via button presses reflecting answers or acceptance.

Communication from RTX337x to Patient
The RTX337x can communicate to the Patient in different scenarios. The communication is written and/or verbal. Text and images can be displayed in the display, and corresponding verbal information can be played by a speaker, see figure 4.

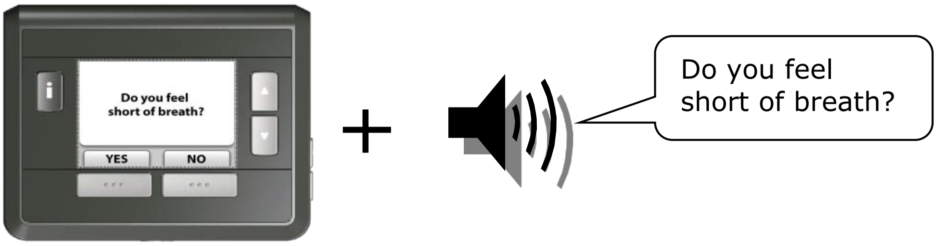


Figure 4. Communication from the RTX337x to the Patient will be written (display) and/or verbal (speaker).

All text, image and audio files are controlled by JavaScripts, see examples in section 10.11. The JavaScripts for Patient communication can be based on a Question Tree structure as seen in the example on figure 5.

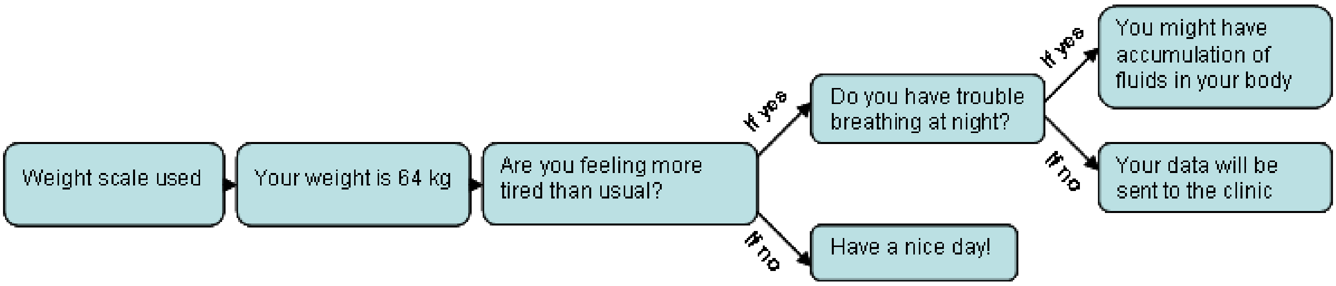

Weight scale used
Your weight is 64 kg
Are you feeling more tired than usual?
If yes
Do you have trouble breathing at night?
If yes
You might have accumulation of fluids in your body
If no
Your data will be sent to the clinic
If no
Have a nice day!

Figure 5. Example of a Question Tree structure for JavaScripts.

An Event (time controlled, use of external device, etc.) will trigger the execution of a JavaScript. The road through the JavaScript can be dependent on the answers given by the Patient.

Communication from Patient to RTX337x

There are five buttons on the front of the RTX337x for direct patient interaction. Two selection (soft) buttons below the display, 'scroll up' and 'scroll down' buttons at the right side of the display and an 'info' button on the left side of the display. Also two buttons for volume 'up' and 'down' control can be found on the side of the RTX337x.

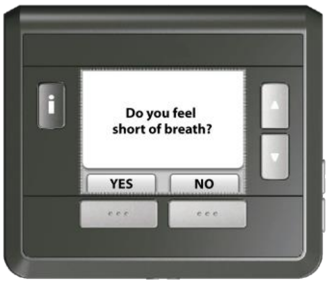


Figure 6. The RTX337x has 5 buttons on the front. Two selection (soft) buttons below the display, scroll up, scroll down on the right of the display and an info button on the left of the display. Two buttons for volume up and down can be found on the side.

The Patient communicates to the RTX337x by answering questions or statements. This is done by pressing the buttons. The answers will be registered and the JavaScripts will decide what happens next.

Communication with external devices
The RTX337x supports a number of external devices that monitors the patient's vital signs. For more information on the external devices and how there are configured, see chapter 13. The communication with the external devices is through Bluetooth, IR or RS232 cable, see figure 7. When an external device has been configured for in an RTX337x (see chapter 13), the data transfer from external devices to an RTX337x is performed automatically.

NOTE! For most of the external devices time will be synchronized with the RTX337x time. Otherwise the time difference between the RTX337x clock and the device clock will be reported.

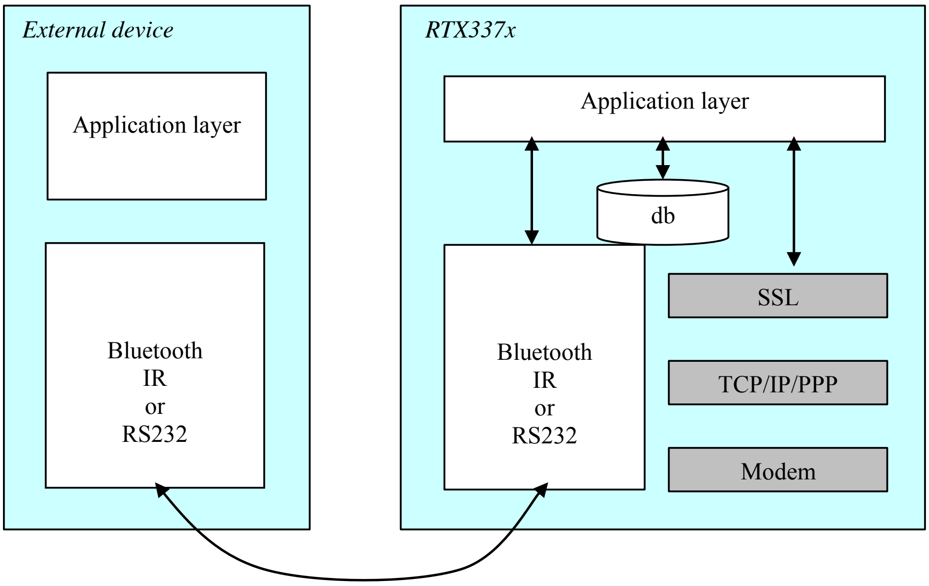


Figure 7. The figure illustrates the communication between external devices and the RTX337x.

5.3 Supervision and Timing

The RTX337x can supervise all external devices to see if the external device is being used within a desired period of time. For example it is possible to set supervision for a weight scale (external device) every day at 9.00 am. If the weight scale does not send any data to the RTX337x between midnight and 9.00 am, the supervision event will occur and if a script is assigned to the event, the script will be activated. If measurement data is sent from the weight scale to the RTX337x at e.g. 8.45 am, no supervision event will occur at 9.00 am that day.
Supervision is set with AT+pDVS, for more information see section 9.1.13.

5.3.1 Virtual Device – internal device

The virtual device is an internal device in the RTX337x. The virtual device can:

- Set a timing event at specified times. AT+pDV is used for this. See section 9.1.12 for more information.
- Over-view server connections. AT+pDVS is used for this.

Installation of the virtual device is described in section 0, Virtuel Device.

Timing (AT+pDV) for the virtual device is used for e.g. prompting messages (medical reminders, measurement requests, etc.) at specific times or time intervals.

Supervision (AT+pDVS) for the virtual device is used for over-viewing whether a server connection has been established within a desired time period. If no server connections have occurred within the period of time, a supervision event will occur. If a script is assigned to the supervision event the script will be activated. The script could for example ask the patient to check phone connections or ask the patient for reasons for the missing server connections.

5.3.2 Device supervision and time settings

If the time is adjusted in the RTX337x the supervisions and timings will behave in the following manor:

1. If the time is set back all supervision and timing events (that are present in the time gab) will occur again at the specified times.
2. If the time is set forward all supervision and timing events (that are present in the time gab) will occur (delayed if they were to appear somewhere in the time gab).

NOTE! Whenever the RTX337x contacts the server the RTX337x clock is synchronized with the server time. This can lead to double appearance of events if the time in the RTX337x is set back.

6 Getting started

6.1 Parts identification

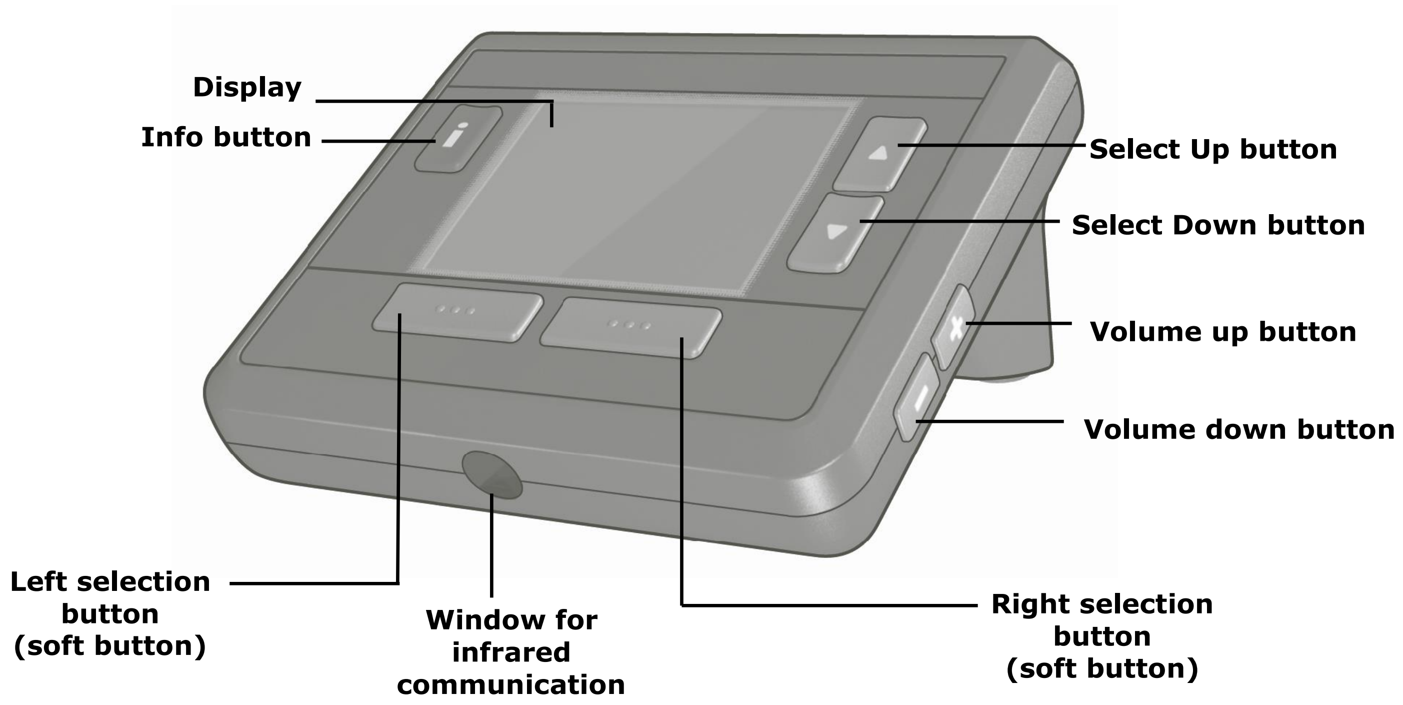

Display
Info button
Select Up button
Select Down button
Volume up button
Volume down button
Left selection button (soft button)
Window for infrared communication
Right selection button (soft button)

Figure 8. The RTX337x front side.

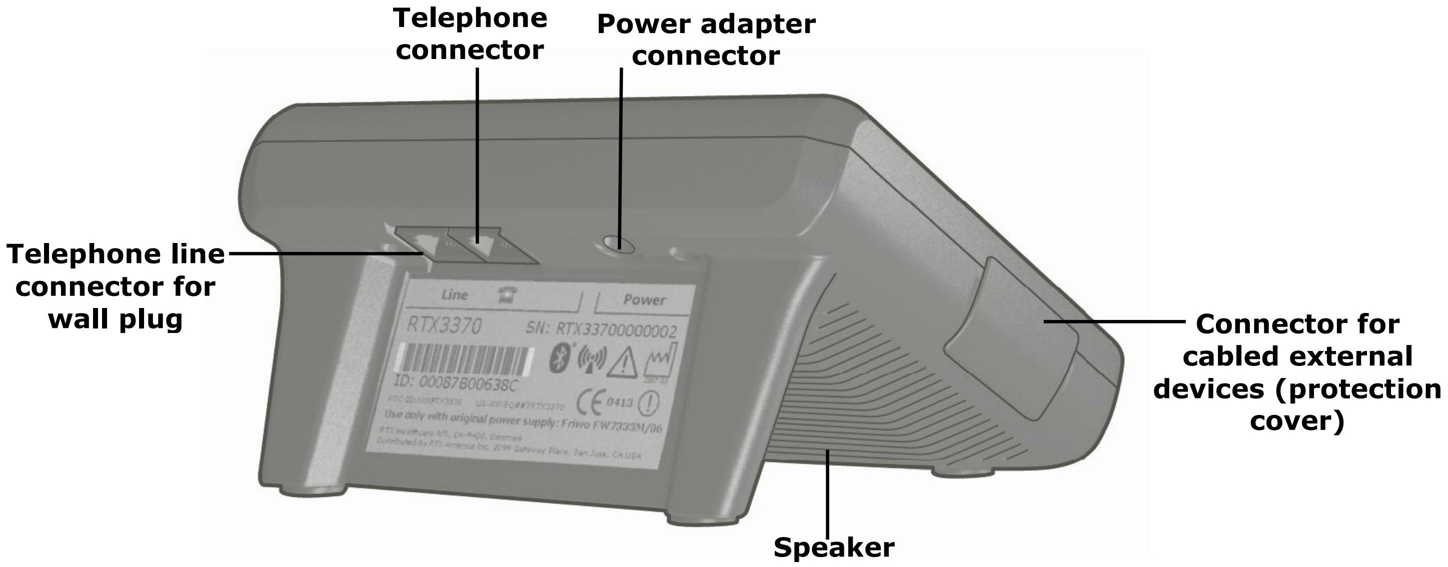

Telephone connector
Power adapter connector
Telephone line connector for wall plug
Connector for cabled external devices (protection cover)
Speaker

Figure 9. The RTX3370 rear side.

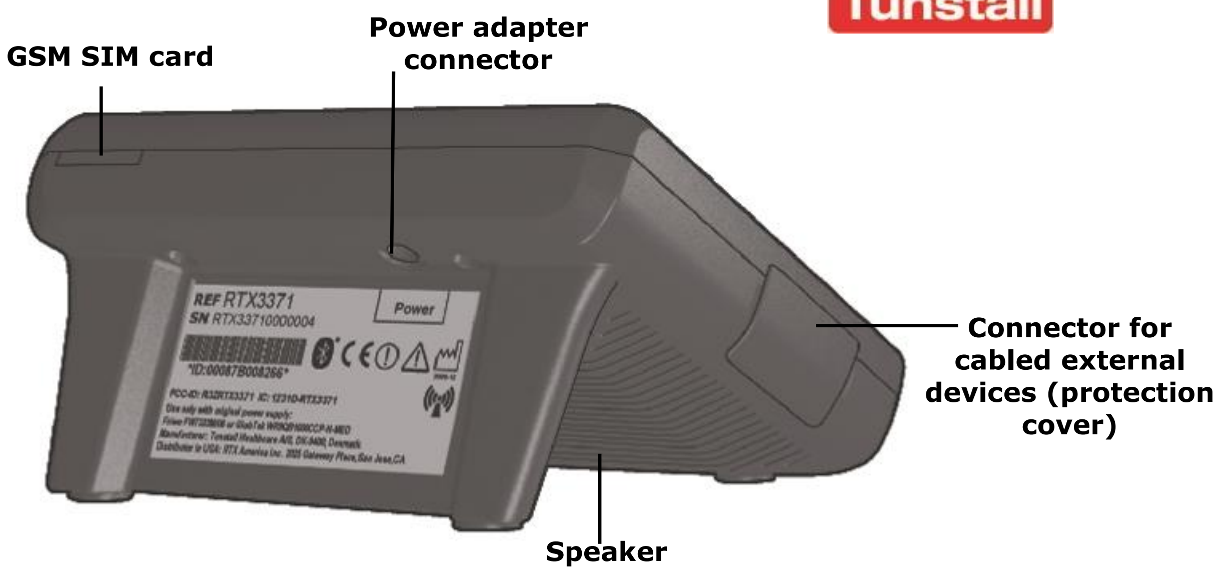


Figure 10. The RTX3371 rear side.

6.2 Installation Step by Step

Before installation, please read all the information provided in this chapter!
Inspect the package for any damage. In the event of damage, please call for service. Always keep the original package for return shipment.

1. ***Connect RTX3370 to the telephone system (RTX3370 only):***

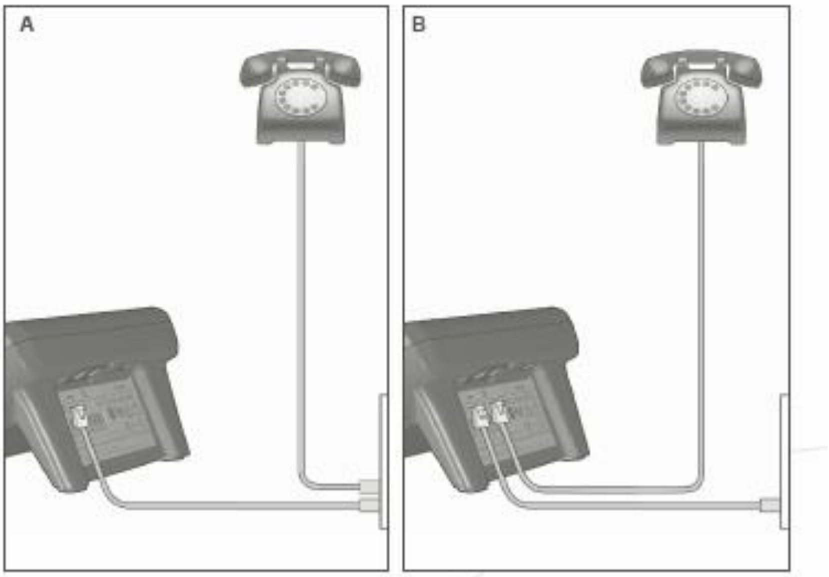


Figure 11. RTX3370 and telephone connected to a double telephone outlet (A). RTX3370 connected to the telephone outlet and the telephone connected to the RTX3370 (B).

If you have a double telephone outlet in the wall, connect both the phone and the RTX337x to the outlet (picture A). If you have a single telephone outlet in the wall, first connect the telephone to the RTX337x, then connect the RTX337x to the outlet (picture B).

It is important that the cables are carefully arranged and secured to avoid unintended disconnection, or the possibility of tripping or blocking the path of a wheelchair or other moving objects.
Be careful not to use a power socket that may be switched off unintendedly. Do not install or service the device, power adapter or cables during an electrical storm.

2. ***Connect the Power supply adapter to an electrical outlet***

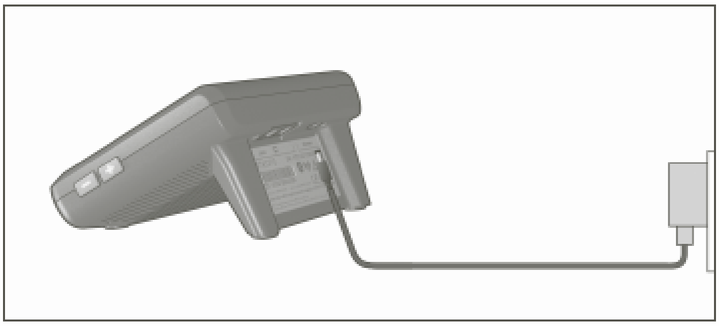


Figure 12. Power supply for the RTX337x connected to an electrical outlet.

IMPORTANT: Never disconnect the power supply adapter from the RTX337x.
Make sure that the power source cannot be switched off unintendedly.

6.3 Connecting the PSTN phone line (RTX3370 only)

The RTX3370 contains a built-in PSTN analogue V32bis data modem. The RTX3370 is designed to be connected to the patient's home phone line in parallel to existing phones, fax etc. The RTX3370 supports a number of features, like the "Line in Use Detection" and "Intrusion Detection" described below. These features ensure that the RTX3370 does not block for emergency calls and other outgoing phone calls.

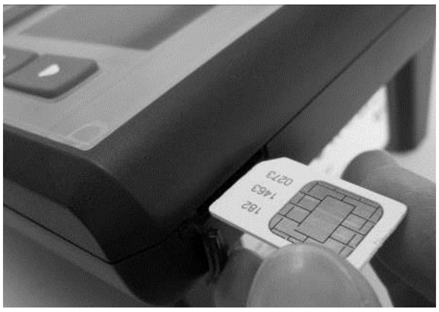
Line in Use Detection
The Line in Use Detection feature stops the modem from disturbing the phone line when the line is already being used.
When the RTX3370 tries to dial and the phone line is in use, the modem in the RTX3370 will not go off hook.

Intrusion Detection
The Intrusion Detection feature (also commonly referred to as Line Pickup Detection or Parallel Phone Detection (PPD)) allows the modem in the RTX3370 to detect when another telephony device i.e. a phone or fax machine is attempting to use the phone line. This feature is used to quickly drop a modem connection in the event that a user picks up an extension phone line, and thereby give normal voice users the highest priority over the RTX3370 usage of the telephone line. If the line is not released hang up and try again after a few seconds.

6.4 Inserting SIM Card (RTX3371 only)

When using an RTX3371 (GSM/GPRS) a SIM Card must be inserted.
If the SIM card it locked with PUK code, the SIM card must be unlocked before inserting it into the RTX3371 (a mobile/cell phone can be used for this). If a PIN Code is used, the PIN code must be given to the RTX3371 with the AT+pPINCODE (see section 9.2.20). Changing the SIM Card requires changing the PIN code with AT+pPINCODE, unless the same PIN code is also valid for the new SIM Card. SIM Card must be directed with

contacts pointing upwards and the ‘cut’ corner going in first. See picture for illustration of SIM Card insertion.



SIM Card must be directed as shown in the picture when inserted to the RTX3371.

7 Configuring the RTX337x

The RTX337x is a programmable device, and needs to be carefully configured in order to provide the optimal performance and system integrity. This includes both setting up the RTX337x configuration variables as well as appropriate JavaScript's to be written and uploaded.

There are two ways to configure and program the RTX337x. Either by using the serial service interface with a terminal program like Windows® HyperTerminal, or by transmitting configuration data over the internet (refer to chapter 11, Web Server Integration).

Hardware needed for configuration:

- RTX337x
- NULL modem serial cable for Service communication interface
- Power supply for the RTX337x
- Computer with an RS-232 port

Step by step guide for configuration via the technical function interface:
For this step by step guide a Windows HyperTerminal is used as example for technical function interface.

1. Open a Windows® HyperTerminal window for TTY communication.
Set the following RS232 communication parameters:

115200 Baud

8	Data bits
N	No Parity
1	Stop bit

Hardware flow control is required on the PC.

NOTE: Not all NULL modem cables have support for the handshaking signals.

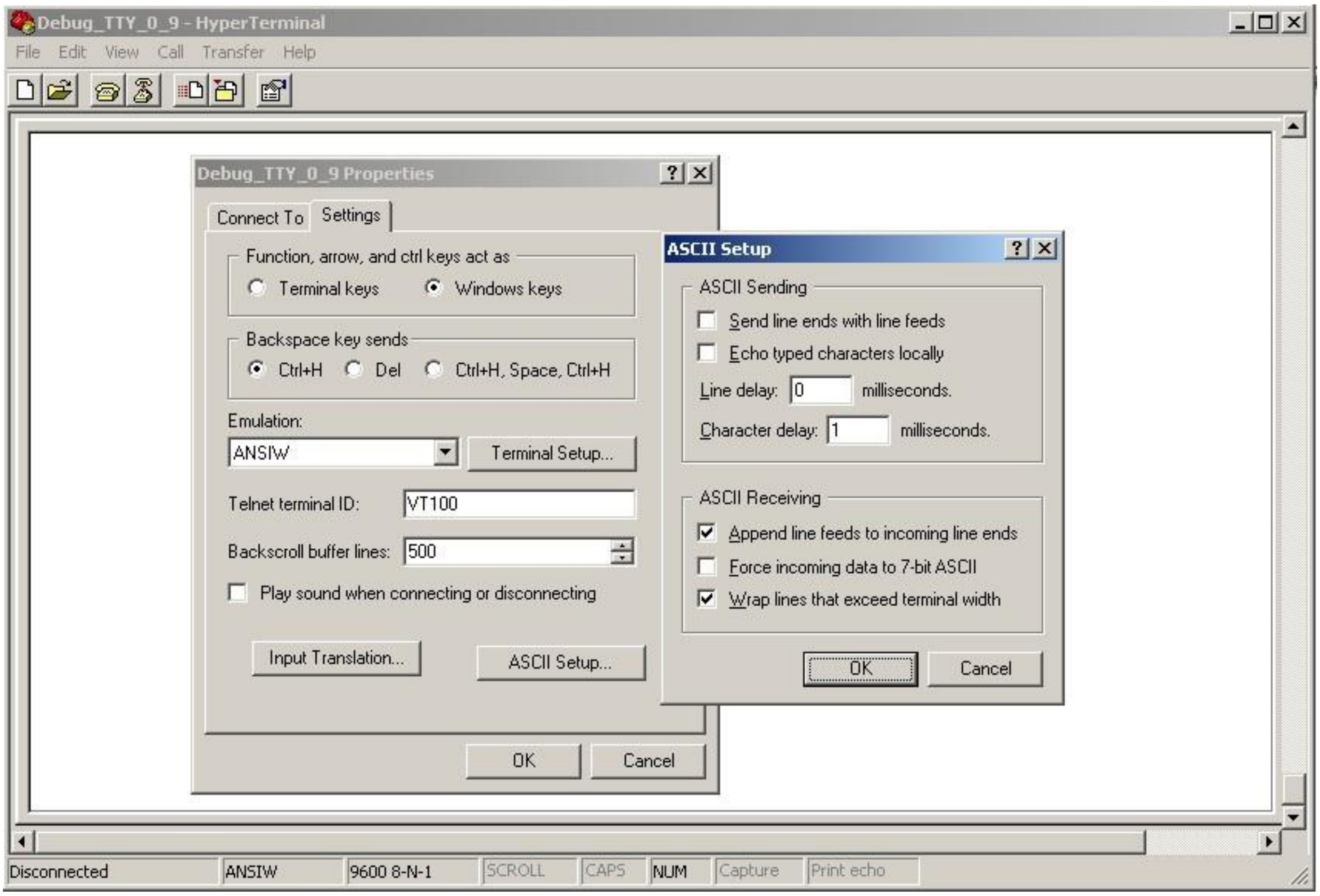
The format of configuration data is based on an extension to the Hayes AT command set (for details refer to chapter 9, AT+p Command Set).

NOTE: Depending on the RTX337x current operating mode as well as the JavaScript configuration the RS-232 connector might be in external device mode and not technical function interface mode. Setting the RTX337x in factory mode causes the interface to always be in technical function interface mode.

To enter configuration commands and/or scripting files, the RTX337x mode has to be changed to FACTORY mode or the technical function interface enabled through JavaScript. FACTORY mode can be entered by pressing the right 'selection button', 'scroll down' and 'volume down' buttons simultaneously for 5 seconds.

NOTE: JavaScript support for enabling the technical function interface will depend on the custom configuration performed.

Additional Setup for Windows® HyperTerminal, see figure 14:



Function, arrow, and ctrl keys act as: ○ Terminal keys ● Windows keys

Backspace key sends: ● Ctrl+H ○ Del ○ Ctrl+H, Space, Ctrl+H

Emulation: ANSIW

Telnet terminal ID: VT100

Backscroll buffer lines: 500

□ Play sound when connecting or disconnecting

ASCII Sending

□ Send line ends with line feeds

□ Echo typed characters locally

Line delay: 0 milliseconds.

Character delay: 1 milliseconds.

ASCII Receiving

☑ Append line feeds to incoming line ends

□ Force incoming data to 7-bit ASCII

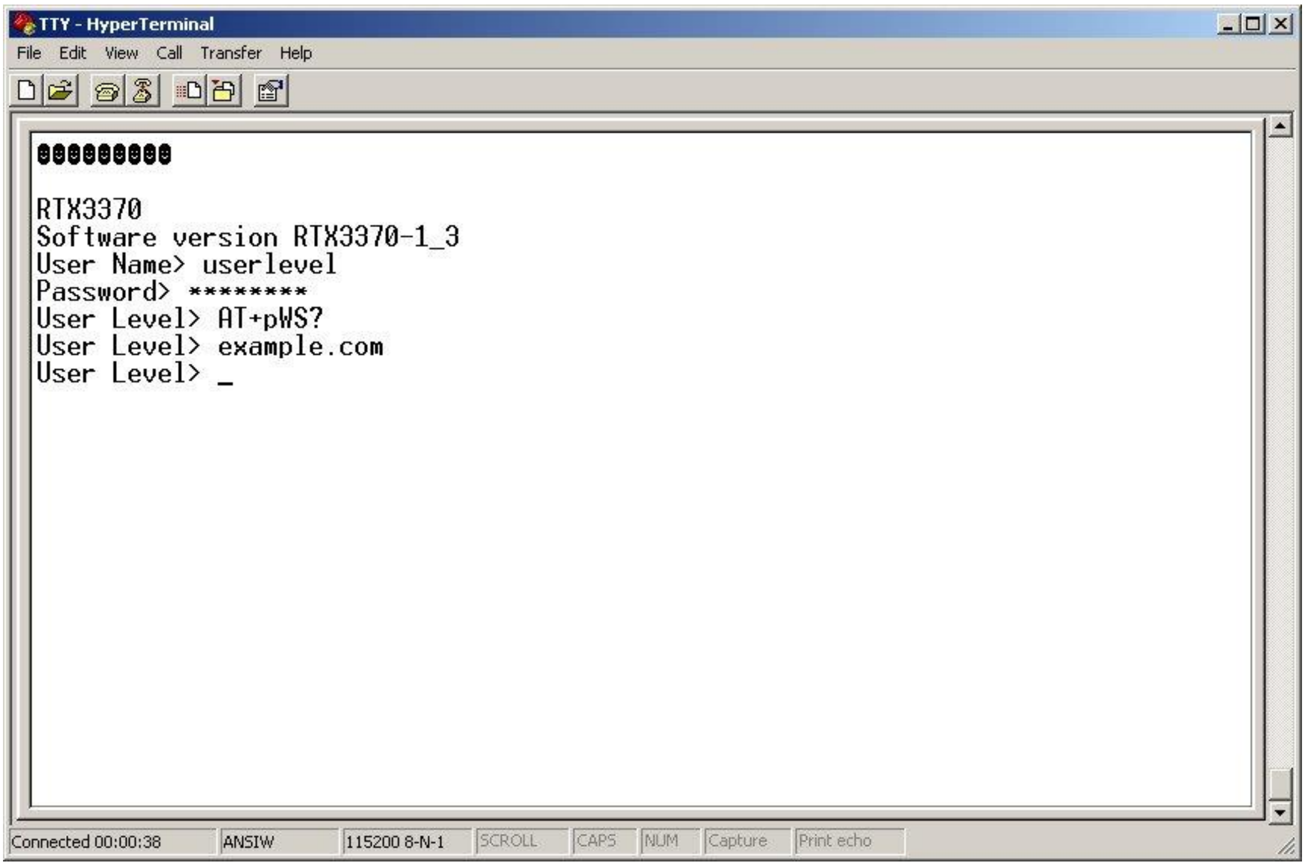
☑ Wrap lines that exceed terminal width

Disconnected | ANSIW | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

Figure 13. Screen shot of the HyperTerminal for technical function interface.

2. After RTX337x power-on, the system login prompt is displayed and the login procedure can be performed, see figure 15.

login>" User level name"[1]
password>" User level password"



```
☻☻☻☻☻☻☻☻☻

RTX3370
Software version RTX3370-1_3
User Name> userlevel
Password> ********
User Level> AT+pWS?
User Level> example.com
User Level> _
```

Figure 14. Screen shot of the HyperTerminal after login and with the AT command AT+pISP?

When entering the username and password, the system responds by changing the prompt to indicate the level of authorization granted.

The default User level username is **userlevel**, and password is **userpass**.

The RTX337x is now ready for configuration, and sending commands to the RTX337x can change the settings.

[1] Access at user level: please refer to section 8.1, Authorizing system login, for further information.

8 Programming AT+p Commands

8.1 Authorizing system login (Username / Password)

Before any configuration can be accomplished, a system login must be performed.

Two levels of system logins are available:

Operator login

Operator login permits access to all commands.

It is recommended that only the system responsible staff is granted the username and password for Operator login.

Factory setting Username = "operlevel"
Factory setting Password = "operpass"

The Operator login username and password can be changed using the AT+pOPER command.

User login

User login permits access to a limited number of commands

It is recommended that in-field and third party employed staff is granted the username and password for the User login only.

Factory setting Username = "userlevel"
Factory setting Password = "userpass"

The User login username and password can be changed using the AT+pUSER command. (This command requires Operator login)

NOTE: The Factory settings may have been changed, if units have been delivered with a custom configuration already downloaded.

The Operator login and User login are single logins, which means that they cannot be treated as personal logins, since they might be shared among several individuals.

Username	The username can consist of any ASCII character excluding Carriage Return (HEX 0D) and colon (HEX 3A). Username is case sensitive.
Password	The Password can consist of any ASCII character excluding Carriage Return (HEX 0D) and colon (HEX 3A). Password is case sensitive.

After entering Factory mode, the system login prompt is displayed and the login procedure can be performed.

login>**"Operator or User level name"<cr>**
password>**"Operator or User level password"<cr>**

After entering the username and password, the system responds by changing the prompt to indicate the level authorization granted. For example:

Operator Level>
        or
User Level>

The RTX337x is now ready for configuration, and sending commands to the RTX337x can change settings.

Commands not known at the actual login level will result in the following result code: **ERROR - Wrong login level or unknown command**

After configuration, the logout procedure must be accomplished using the logout command: (this example is assuming a User level login)

User Level>**logout<cr>**

If the RTX337x is *not* in Factory mode, the RS232 port will after logout be activated for external device interface.

8.1.1 Direct Connection

The configuration is accessible by direct connection via the RS232 connector.

Use a serial terminal program like HyperTerminal. If binary data needs to be uploaded as part of the configuration, the program must support Ymodem. The AT+pBINARYUPLOAD command must be used to put the system in Ymodem download mode. The binary data must have the format described under the AT+pBINARYUPLOAD command in section 9.1.17.

Commands are executed by the system as batches. A batch is defined as either a single command or a series of commands between an AT+pLCK command and an AT+pUNLCK command. Any binary data transferred through Ymodem is buffered and must be used in the next batch of commands. The buffer is cleared of any content at the end of the next batch.

8.1.1.1 Example Session

The following is an example session, from login to logout:

```
login>operlevel<cr>
password>operpass<cr>
Operator Level>AT+pFS=<cr>
OK
Operator Level>logout<cr>
```

Example session containing binary data and executed as a batch:

```
login>operlevel<cr>
password>operpass<cr>
Operator Level>AT+pBINARYUPLOAD<cr>
Activating Ymodem mode
Please start uploading or wait for timeout
CC (Start upload in terminal program)
Data received, buffering
Leaving Ymodem mode
Operator Level>AT+pLCK<cr>
```

```
      OK
      Operator Level>AT+pINSVOICE=voice1<cr>
      OK
      Operator Level>AT+pINSSCRIPT=script1<cr>
      OK
      Operator Level>AT+pUNLCK<cr>
      OK
      --Buffered binary data cleared--
Operator Level>logout<cr>
```

8.1.2 Network Connection

The RTX337x can be configured via a Web Server interface as described in section 12.4. When the RTX337X sends a REQUEST type telegram, the server can reply with a CONFIG type telegram. In this telegram the configuration commands is in the <Value> tag of the XML and binary data is attached as described in section 12.3.

Example CONFIG type telegram from server showing insertion of a JavaScript. The JavaScript itself is attached as binary data.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xml xmlns="RTX-H#1-Server">
      <Gateway>
            <GatewayId>00087B0063B6</GatewayId>
            <Time>1182248687</Time>
            <GwChksum>875365522</GwChksum>
      </Gateway>
      <Monitor>
            <Type>CONFIG</Type>
            <Value>
                  operlevel
                  operpass
                  AT+pLCK
                  AT+pINSSCRIPT=QTV8.js
                  AT+pUNLCK
L                 logout
            </Value>
            <DvChksum>3191215927</DvChksum>
      </Monitor>
</xml>
```

The RTX337X responds with a RESPONSE type telegram:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xml xmlns="RTX-H#1-Client">
      <Gateway>
            <Time>1182248692</Time>
            <GatewayId>00087B0063B6</GatewayId>
            <GwChksum>4282124932</GwChksum>
            <GwInfo>50100002, XX, RTX337x-1_2</GwInfo>
            <MessageNumber>48</MessageNumber>
      </Gateway>
      <Monitor>
            <Value>
                  User Name&gt; operlevel
                  Password&gt; operpass
                  Operator Level&gt; AT+pLCK
                  Operator Level&gt; OK
                  Operator Level&gt; AT+pINSSCRIPT=QTV8.js
                  Operator Level&gt; OK
                  Operator Level&gt; AT+pUNLCK
                  Operator Level&gt; OK
                  Operator Level&gt; logout
                  User Name&gt;
            </Value>
            <DvChksum>2371439844</DvChksum>
            <Type>RESPONSE</Type>
      </Monitor>
</xml>
```

NOTE: the logout command must be the last command when used from the server, since two logins in the same CONFIG telegram is not allowed.

9 AT+p Command Set

This section describes and lists all commands available for configuring the RTX337x.
The commands are divided in five categories, listed by their functions.
The five categories cover commands for:

- **Configuration**
- **Communication**
- **Security**
- **Service**
- **Testing**

The format of commands is based on an extension to the Hayes modem AT command set. The AT+p command set contains a number of proprietary AT commands available to configure and set-up all programmable parameters in the RTX337x.

The AT+p command body is restricted to printable ASCII characters (032 – 122). The command terminator is the ASCII <CR> (carriage return) character. The command line interpretation begins upon receipt of the carriage return character. If a syntax error is found anywhere in a command line, the line will be ignored and an error message will be returned.

NOTE: The command interpreter is case sensitive.

The following table is a summary of available commands. See the individual commands for details:

	Command	Level	Description
Configuration	AT+pLCK	User	Lock Updates
	AT+pUNLCK	User	Unlock Updates
	AT+pTIME	User	Set or request System Time & Date
	AT+pGWID	User	Request Gateway Identification
	AT+pCODE	Operator	Set or request the current product CODE
	AT+pOPER	Operator	Set or request Operator Identification
	AT+pUSER	Operator	Set or request User Identification
	AT+pAUTODIAL	User	Set or request the Autodial setting.
	AT+pGW	User	Set or request Gateway Timing
	AT+pGWR	User	Set or request Gateway Retry Timing
	AT+pDV	User	Set or request Device Timing
	AT+pDVS	User	Set or request Device Supervisor
	AT+DVLIMITFIRST	User	Set or request Device measurement Limit for First connection.
	AT+pINSDV	User	Insert Device or list Devices
	AT+pRMDV	User	Remove Device
	AT+pUPDDV	User	Update Device
	AT+pINSFONT	User	Insert or List TrueType fonts
	AT+pRMFONT	User	Remove Font
	AT+pFONTASSIGN	Operator	Set or request assignment of Truetype fonts
	AT+pFONTSIZE	Operator	Set or request TrueType font sizes
	AT+pBINARYUPLOAD	User	Put the TTY in Ymodem upload mode
	AT+pINSVOICE	User	Insert Phrase or list Phrases
	AT+pRMVOICE	User	Remove Phrase
	AT+pSCRIPTSTOR	User	Set or request Javascript storage values
	AT+pINSSCRIPT	User	Insert script or list Scripts
	AT+pRMSCRIPT	User	Remove script
	AT+pINSIMAGE	User	Insert image or list image
	AT+pRMIMAGE	User	Remove image
	AT+pINSFONT	User	Insert TrueType font file to the database

	AT+pRMFONT	User	Removes font files from the database
	AT+pFONTASSIGN	Operator	Sets the assignment of TrueType fonts
	AT+pFONTSIZE	Operator	Sets the size of the TrueType
	AT+pBACKGROUND	Operator	Set or request background image configuration
	AT+pMODE	Operator	Set or request operation mode
	AT+pFACTORYSCRIPT	Operator	Set or request FACTORY mode script name
	AT+pPATIENTSCRIPT	Operator	Set or request PATIENT mode script name
	AT+pNORMALSCRIPT	Operator	Set or request NORMAL mode script name
	AT+pTECHSCRIPT	Operator	Set or request technical menu script name
	AT+pBOOTSCRIPT	Operator	Set or list Boot script name
	AT+pVOLUME	User	Set or request volume configuration
	AT+pBACKLIGHT	User	Set or request backlight configuration
	AT+pBTLOCALNAME	User	Set or request
Communication	AT+pISP	User	Set or request ISP phone number
	AT+pISP_BACKUP	User	Set or request backup ISP phone number
	AT+pISP_VC	User	Set or request ISP Phone number vector
	AT+pMBEG1	User	Set or request init string for modem
	AT+pMBEG1_VC	User	Set or request init string vector
	AT+pMBEG2	User	Set or request init string for modem
	AT+pMBEG2_VC	User	Set or request init string vector
	AT+pMCDBEG	User	Set or request control dial string for modem
	AT+pMCDBEG_VC	User	Set or request control dial string vector
	AT+pMPRED1	User	Set or request first pre dial string for modem
	AT+pMPRED1_VC	User	Set or request pre dial string vector
	AT+pMPRED2	User	Set or request second pre dial string for modem
	AT+pMPRED2_VC	User	Set or request pre dial string vector
	AT+pMPRED3	User	Set or request third pre dial string for modem
	AT+pMPRED3_VC	User	Set or request pre dial string vector
	AT+pMEND	User	Set or request clean up string for modem
	AT+pMEND_VC	User	Set or request clean up string vector
	AT+pMCDEND	User	Set or request control dial string for modem
	AT+pMCDEND_VC	User	Set or request control dial string vector
	AT+pPINCODE	User	Set or request GSM pin code
	AT+pGPRSAPN	User	Set or request GPRS access point name
	AT+pGPRSIP	User	Set or request IP address applicable to the GPRS PDP
	AT+pGPRSHEADCOMP	User	Set or request GPRS PDP Header compression
	AT+pGPRSDATACOMP	User	Set or request GPRS PDP Data compression
	AT+pSIGNAL	User	Request GSM Signal Strength
	AT+pIMEI	User	Request GSM Module Number
	AT+pWS	Operator	Set or request Web Server IP Address
	AT+pWSPATH	User	Set or request Web Server Path
	AT+pUSR	User	Set or request ISP or GPRS User Name
	AT+pUSR_BACKUP	User	Set or request ISP or GPRS Backup User Name
	AT+pPSW	User	Set or request ISP or GPRS User Password
	AT+pPSW_BACKUP	User	Set or request ISP or GPRS Backup User Password
	AT+pDNS	Operator	Set or request the current DNS server setting.
	AT+pPRXY	User	Set or request Proxy IP Address
	AT+pGZIPCOMP	User	Set or request GZIP compression
	AT+pFALLBACK	User	Allow fallback to last validated settings
	AT+pMTUSIZE	User	Sets the MTU size
	AT+pFORCEMTOMREQ	User	Force the server to send binary data as MTOM attachments
	AT+pCOD	Operator	Set or request Class of Device setting
Security	AT+pSAUT	Operator	Set or request SSL Server Authentication
	AT+pCAUT	Operator	Set or request SSL Client Authentication
	AT+pCACRT	Operator	Set or request SSL CA Certificate
	AT+pCCRT	Operator	Set or request SSL Client Certificate
Service	AT+pVER	User	Request software version

	AT+pLFLG	Operator	Set or request the logging flag
	AT+pFS	Operator	Reset to default or query Settings
	AT+pCLRMESS	Operator	Clear the message buffer
	AT+pTEST	User	Send Test Message
	AT+pLOGSIZE	Operator	
	AT+pLOG	Operator	Send Log to console
	AT+pRFU	Operator	Remote Firmware Update
	AT+pREQFU	Operator	Request a Firmware Update from the server
	AT+pFIRMLIST	Operator	Update the list of approved hardware/software
	AT+pFIRMVALID	Operator	Verify that a firmware is valid on this hardware
	AT+pFIRMCRC	Operator	Set or request CRC and length of the firmware
	AT+pSCRIPT	Operator	Get the current configuration in script form
	AT+pACTIVATE	User	Start script
	AT+pRESETCOUNT	Operator	Set or request the reset count of the device
	AT+pTIMEBACKUP	User	Get the current backed up timestamp
	AT+pSTATUS	User	Get RTX337x status word
	AT+pHOSTSTATUS	User	Get Host status word
	AT+pLASTMEAS	User	Get the last measurement
	AT+pLASTRESP	User	Get the user response to the last measurement
	AT+pTTYENABLE	User	Get the current TTY mode
	AT+pPRODINFO	User	Report production information
	AT+pMODEL	User	Get the hardware model
Testing	AT+pSTOP_SYSTEM	Operator	Stop the system and enter testing mode
	AT+pSTART_SYSTEM	Operator	Restart the system from test mode
	AT+pTEST_BUTTON	Operator	Test buttons and switches
	AT+pMODEM	Operator	Connect directly to the modem for testing
	AT+pSTOP_WATCHDOG	Operator	Stop the watchdog timer for testing
	AT+pBACKLIGHTTST	Operator	Test the LCD backlight
	AT+pDISPLAY	Operator	Display a test pattern on the LCD screen
	AT+pIRDATEST	Operator	Test IrDa communication

9.1 Commands for RTX337x Configuration

9.1.1 AT+pLCK– Lock Updates

The Lock Updates command puts the RTX337x into a mode where a series of AT commands can be entered one at the time, without the individual AT commands taking effect immediately. This may be needed to prevent malfunction and loss of connection during remote operation. This command is also used to indicate the beginning of a batch of commands, used to indicate that more commands are following so buffered binary data should stay in memory.

After the series of AT commands has been entered, the AT+pUNLCK – Unlock Updates command can be entered to allow the commands to take effect. Output (result codes) from each AT command will be seen immediately, but the new values will not be put into effect until the AT+pUNLCK command is issued.

Syntax:	**AT+pLCK**
Parameters:	None
Default value:	None.
Scope:	User level
Result code:	**OK** if not already locked. Otherwise **ERROR – Already locked**.
Example:	**AT+pLCK<cr>** OK

9.1.2 AT+pUNLCK – Unlock Updates

Unlock updates made since updates have been locked (see above) and make the changes conditionally available. This command is also used to indicate the end of a batch, indicating to the system that any buffered binary data shall be deleted.

Syntax:	**AT+pUNLCK**
Parameters:	None.
Default value:	None.
Scope:	User level
Result code:	**OK** if locked. Otherwise **ERROR – Not locked**.
Example:	**AT+pUNLCK<cr>** OK

9.1.3 AT+pTIME – System Time & Date

This command sets the internal system time (Real Time Clock) of the RTX337x. Note that the RTX337x clock is automatically synchronized with the server at every connection and that this command also sets the backup time.

Syntax:	**AT+pTIME=str**
Parameters:	YYYY/MM/DD HH.MM.SS (24-hour clock)
Default value:	None.
Scope:	User level
Result code:	**OK** if syntax is valid. Otherwise **ERROR** returns describing the syntax error.
Example:	**AT+pTIME=2003/01/15 14.33.00<cr>** OK *Sets the time to 14:33:00 (2:33pm) and the date to the 15th of January 2003.*

Report the current time setting

Syntax:	**AT+pTIME?**
Scope:	User level
Result code:	2004/08/09 12.00.00 (example)
Example:	**AT+pTIME?<cr>** 2004/08/09 12.00.00 *Reports current date and time.*

9.1.4 AT+pGWID – Gateway Identification

Report the current value of the RTX337x identification.

Syntax:	**AT+pGWID?**
Parameters:	
Default value:	The GWID is automatically detected from the hardware.
Scope:	User level
Result code:	00A096089A10 (example)
Example:	**AT+pGWID?<cr>** 00A096089A10

9.1.5 AT+pCODE – System code

This command sets the system code, which can be used to determine which restricted devices are allowed to be used on the system. Codes enabling/disabling devices can be obtained from Tunstall.

Syntax:	**AT+pCODE=str**
Parameters:	Str < 200 chars
Default value:	None.
Scope:	Operator level
Result code:	**OK** if syntax is valid. Otherwise **ERROR SAVING DATA IN DATABASE.**
Example:	**AT+pCODE=3030303030303030303730303030303030525 458204865616C74686361726520412F533337343735731<cr>** OK

Report the current permission code.

Syntax:	**AT+pCODE?**
Scope:	User level
Result code:	The current system code.
Example:	**AT+pCODE?<cr>** Permission code: 0X0000000000003030 Customer: Tunstall Healthcare A/S

9.1.6 AT+pOPER – Operator Identification

Sets the operator identification and password.

Syntax:	**AT+pOPER=str** Length of str < 50 characters
Parameters:	str = operator identification (username:password). Both username and password must be at least one character and cannot contain carriage return (hex 0D) or colon (hex 3A). “{“ and “}” characters cannot be used when the command is used remotely. Username and password are case sensitive.
Default value:	operlevel:operpass
Scope:	Operator level
Result code:	**OK** if legal operator identification. Otherwise: **Syntax Error**.
Example:	**AT+pOPER=operlevel:operpass<cr>** OK

Report the current value of the operator identification and password.

Syntax:	**AT+pOPER?**
Scope:	Operator level
Result code:	operlevel:operpass (example)
Example:	**AT+OPER?<cr>** operlevel:operpass

9.1.7 AT+pUSER – User Identification

Sets the user identification and password.

Syntax:	**AT+pUSER=str** Length of str < 50 characters
Parameters:	str = user identification (username:password). Both username and password must be at least one character and cannot contain carriage return (hex 0D) or colon (hex 3A). “{” and “}” characters cannot be used when the command is used remotely. Username and password are case sensitive.
Default value:	userlevel:userpass
Scope:	Operator level
Result code:	**OK** if legal user identification. Otherwise: **Syntax Error**.
Example:	**AT+pUSER=userlevel:userpass<cr>** OK

Report the current value of the user identification and password.

Syntax:	**AT+pUSER?**
Scope:	Operator level
Result code:	userlevel:userpass (example)
Example:	**AT+USER?<cr>** userlevel:userpass

9.1.8 AT+pAUTODIAL – Automatic Dialing

Controls which events make the RTX337x dial the host server automatically.

Syntax:	**AT+pAUTODIAL=str**
Parameters:	str = 1 ⇔ do automatic dialing as dictated by AT+pGW and AT+pGWR. str = 0 ⇔ do not use automatic dialing, only JavaScripts and error conditions can trigger automatic dialing.
Default value:	1
Scope:	User level
Result code:	**OK** if legal value. Otherwise: **Syntax error - Value must be 0 or 1**.
Example:	**AT+pAUTODIAL=1<cr>** OK

Report the current value of Autodial.

Syntax:	**AT+pAUTODIAL?**
Scope:	User level
Result code:	0 (example)
Example:	**AT+pAUTODIAL?<cr>** 0

9.1.9 AT+pGW – Set or request RTX337x Timing

Set or request the timing schedule for contact to the Internet server.

Syntax:	**AT+pGW=str**
Parameters:	Str represents a timing schedule. The Internet server is contacted in accordance with the timing schedule (if the RTX337x has messages queued for the server). **Note:** If configured for interval timing, the interval timer is reset when first message is queued after last timeout. The timing schedule consists of a number of options separated with the character ";". Maximum one option of each kind is allowed. The F and I options excludes each other (only one is allowed). The following options can be used: S-<year.month.day> The timing schedule starts on the given day (start date). The start date consists of max. three numbers separated with the character ".". The three numbers are "year", "month" and "day". Default values represent the current date. If no number is present, then start date uses default values for "year", "month" and "day". If one number is present, then default values are used for "year" and "month", and the number is used for "day". If two numbers are present, then default value is used for "year", and the first number is used for "month", and the second number is used for "day". If three numbers are present, then the first number is used for "year", the second number is used for "month", and the third number is used for "day". N-<number> The timing schedule is active for the given number of cycles. If "number" is zero, then there is no limit on the number of cycles. The default value of the number is zero. When the timing schedule runs

	out, it reverts to event controlled (i.e. immediate). F-<hour.min.sec>-<hour.min.sec>- ….. The timing schedule is represented by the given times (hour.min.sec). The times must be in sequence and within the interval 00:00 to 24:00. Max 10 time settings are allowed. The time indications consist of maximum three numbers separated with the character ".". Default values are zero for "hour", "min", and "sec". If no number is present, then default values (zero) for "hour", "min" and "sec" are used. If one number is present, then default values (zero) are used for "hour" and "min", and the number is used for "sec". If two numbers are present, then default value (zero) is used for "hour", and the first number is used for "min", and the second number is used for "sec". If three numbers are present, then the first number is used for "hour", the second number is used for "min", and the third number is used for "sec". If the F option is used, then this excludes the I option. I-<hour.min.sec> The timing schedule is represented by the given time interval (hour.min.sec). The format is the same as described above for the F option. In case this interval is 0, then the timing schedule is event controlled (i.e. immediate). If the I option is used, then this excludes the F option.
Default value:	S- ; N- ; I- i.e. current date for the start date, no limitation on the number of cycles, and event controlled timing schedule.
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise: **Syntax Error**.
Example:	**AT+pGW=I-0<cr>** OK

Report the current value of the RTX337x timing.

Syntax:	**AT+pGW?**
Scope:	User level
Result code:	I-0 (example)
Example:	**AT+pGW?<cr>** I-0

9.1.10 AT+pGWR – Set or request RTX337x Retry Timing

Set or request the RTX337x (retry) timing schedule for contact to the Internet Server.

Syntax:	**AT+pGWR=str**
Parameters:	Str represents a (retry) timing schedule. The Internet server is contacted in accordance with this timing schedule (if the RTX337x has messages queued for the server). This timing schedule is used in cases where the RTX337x by some reason has decided that the normal timing schedule (see 9.1.8) should be replaced (no contact with server). If set to I-0 this function is disabled. Format of the timing schedule is described in 9.1.9.
Default value:	I-0 (disabled)
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise: **Syntax Error**.
Example:	**AT+pGWR=F-00.30.00-01.00.00<cr>** OK

Report the current value of the retry timing schedule.

Syntax:	**AT+pGWR?**
Scope:	User level
Result code:	F-00.30.00-01.00.00 (example)
Example:	**AT+pGWR?<cr>** F-00.30.00-01.00.00

9.1.11 AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection

Set or request device measurement limit for first connection.

Syntax:	**AT+pDVLIMITFIRST=str**
Parameters:	str=<deviceid>:<limit> A maximum limit (<limit>) of received measurements at first connection from a specific device (identified by <deviceid>) can be set. If <limit> is set to 0 there is no limitation of measurements.
Default value:	<deviceid>:0, meaning no limit of measurements at first connection.
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise: **Syntax error - Value must be >= 0**.
Example:	**AT+pDVLIMITFIRST=Dev1:5<cr>** OK
Note:	This command cannot be used in the same configuration telegram from the server as the AT+pINSDV command. Instead use the AT+pINSDV command to set the measurement Limit for First connection This function is only functional for some devices please see section 13 - Installation of External devices.

Report the current values of supervision formats for all devices.

Syntax:	**AT+pDVLIMITFIRST?**
Scope:	User level
Result code:	Dev1:5 (example)

9.1.12 AT+pDV – Set or request Device Timing

Set or request timing schedule for contact to a given device.

Syntax:	**AT+pDV=str**
Parameters:	Format is <devicename>:<timeschedule> Format of <timeschedule> is described in 9.1.9. The device (identified by <devicename>) is polled in accordance with the given time schedule.
Default value:	<devicename>:I-0. Contact to device is event controlled (no poll).
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise: **Syntax Error**.
Example:	**AT+pDV=Dev1:I-0<cr>** OK
Note:	This command cannot be used in the same configuration telegram from the server as the AT+pINSDV command. Instead use the AT+pINSDV command to set the timeschedule.

Report the current values of the timing schedules for all devices.

Syntax:	**AT+pDV?**
Scope:	User level
Result code:	Dev1:I-0 (example)

9.1.13 AT+pDVS – Set or request Device Supervisor

Set or request supervision of a given device.

Syntax:	**AT+pDVS=str**
Parameters:	Format is <devicename>:<timeschedule> Format of <timeschedule> is described in 9.1.9. Devices can be supervised. It is supervised that the device (identified by <devicename> has been used according to the schedule. This includes either x hours since last usage or no usage in last interval on a time schedule with fixed times. A notification is raised if the device has not been used according to this specification.
Default value:	<devicename>:I-0, meaning that the device is not supervised.
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise: **Syntax Error**.
Example:	**AT+pDVS=Dev1:I-0<cr>** OK
Note:	This command cannot be used in the same configuration telegram from the server as the AT+pINSDV command. Instead use the AT+pINSDV command to set the timeschedule.

Report the current values of supervision formats for all devices.

Syntax:	**AT+pDVS?**
Scope:	User level
Result code:	Dev1:I-0 (example)

9.1.14 AT+pINSDV – Insert or List Devices

Insert device into the RTX337x.

Syntax:	**AT+pINSDV=str**
Parameters:	str=<deviceid>:<devicetype>:<deviceinfo>[:<devicetiming>[:<devicesupervise>[:<Devicelimitfirst>]]] Devicename: An ASCII string of max. 30 characters (letters or digits). This information identifies the device and must be unique. Devicetype: Length of devicetype < 20 characters. This information determines the type of device attached. Deviceinfo: Length of deviceinfo < 200 characters. For details about the deviceinfo string for each device, please see chapter 13, Installation of External Devices. Devicetiming (optional): Length of devicetiming < 200 characters. Configures the timing schedule. This schedule is for some devices used to poll for contact with the device. Please see section 9.1.12, AT+pDV, for details. Devicesupervise (optional): Length of devicesupervise < 200 characters. Configures the supervision schedule. Please see section 9.1.13, AT+pDVS, for details. Devicelimitfirst (optional) Length of Devicelimitfirst < 20 characters. Configures the limit of measurements at first connection. Note this function is only functional for some devices please see section 13 - Installation of External devices.
Default value:	
Scope:	User level
Result code:	**OK** if device could be inserted **Syntax Error** if missing colons. **ERROR SAVING DATA IN DATABASE** if devicename already exists. **Syntax Error. Illegal device name.** If illegal device name. **Device "Devicetype" not permitted** if the current system code does not allow this device. **ERROR - Only one generic Bluetooth device permitted** if a generic Bluetooth device already exists. **ERROR - Only one IR device permitted** if an IR device already exists. **ERROR - Only one RS232 device permitted** if an RS232 device already exists. The devicetype and deviceinfo is NOT validated beside the string length, and care should be taken to assure correct data.
Example:	**AT+pINSDV=Dev1:ADBTWSType:00043EC204D2-39121440<cr>**

	OK **AT+pINSDV=Dev3:LsBgmType:RST89A6QT<cr>** OK

Report a list of all inserted devices.

Syntax:	**AT+pINSDV?**
Scope:	User level
Result code:	List of all devices.

9.1.15 AT+pRMDV – Remove Device

Removes device from the RTX337x.

Syntax:	**AT+pRMDV=str**
Parameters:	str=<devicename> Length of str < 30 characters.
Default value:	
Scope:	User level
Result code:	**OK** if device existed, otherwise **DEVICE NOT IN DATABASE** **Syntax Error. Illegal device name.** If device name too long.
Example:	**AT+pRMDV=Dev1<cr>** OK

9.1.16 AT+pUPDDV – Update Device

Update device configuration in the RTX337x.

Syntax:	**AT+pUPDDV=str**
Parameters:	str=<devicename>:<deviceinfo> Devicename: Length of devicename < 30 characters. This information identifies the device and must be unique. Deviceinfo: Length of deviceinfo < 200 characters. For Details about the Deviceinfo string please see chapter 13, Installation of External Devices.
Default value:	
Scope:	User level
Result code:	**OK** if device could be updated **Syntax Error** if missing colons. **ERROR SAVING DATA IN DATABASE** if devicename does not exists. The deviceinfo is NOT validated beside the string length, and care should be taken to assure correct data.
Example:	**AT+pUPDDV=Dev1:00043EC204D2-39121440<cr>** OK **AT+pUPDDV=Dev2:B011334B<cr>** OK **AT+pUPDDV=Dev3:RST89A6QT<cr>** OK
Note:	This command cannot be used in the same configuration telegram from the server as the AT+pINSDV command. Instead use the AT+pINSDV command to set deviceinfo.

9.1.17 AT+pBINARYUPLOAD – Put the TTY in Ymodem upload mode

This puts the technical function interface into Ymodem mode. It will stay in this mode until it times out or an upload completes.

Syntax:	**AT+pBINARYUPLOAD**
Parameters:	None.
Default value:	None.
Scope:	User level
Result code:	See example below
Example:	Operator Level>**AT+pBINARYUPLOAD<cr>** Activating Ymodem mode Please start uploading or wait for timeout CC Data received, buffering Leaving Ymodem mode Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSVOICE=voice1<cr>** OK Operator Level>**AT+pINSVOICE?<cr>** A 20 voice1 1024 1 KiB used by this type, 4350 KiB unused memory. Operator Level>**AT+pUNLCK<cr>** OK **--Buffered binary data cleared--**
Note:	It is important that this command is not between an AT+pLCK and an AT+pUNLCK command. The command is also only valid from the technical function interface, *not* from the server. The contents of the uploaded data must be in the format described in section 10.3, Uploading JavaScripts, sound files and images.

9.1.18 AT+pINSVOICE – Insert or List Voice

The Insert Voice command adds phrase files to the database.

Syntax:	**AT+pINSVOICE=str**
Parameters:	An ASCII string of max. 10 characters (letters or digits). This string is the name for identifying the accompanying binary data as described in section 10.3, Uploading JavaScripts, sound files and images.
Default value:	None.
Scope:	User level
Result code:	**OK** if the voice is successfully stored, otherwise **ERROR SAVING DATA IN DATABASE**
Example:	**--Binary file is transmitted and buffered—** Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSVOICE=voice1<cr>** OK Operator Level>**AT+pUNLCK<cr>** OK **--Buffered binary data cleared--**
Note:	If a phrase file with the same name exists on the system it will be replaced.

The format of the content will consist of two sub records:

a. A SBC—file with the sound track (speech).
b. A txt string with the UTF-8(for TrueType font) or Latin-1(for build in font) text for the above SBC file

The format of the content will be:
A header consisting of 2 fields of 4 Bytes (Little Endian format) each containing the length in bytes of each of the sub records followed by the 2 records in the order a, b.

The name (parameter for the command) will be used by the system for identifying the phrase in the binary data accompanying the configuration batch. The name will also be used in question trees for referencing the phrase.

Report a list of all phrase files.

Syntax:	**AT+pINSVOICE?**
Scope:	User level
Result code:	List of all phrase files, and their size. Also prints a status of used and unused storage.
Note:	If a phrase file with the same name exists on the system it will be replaced.
Example:	Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSVOICE?<cr>** A 20 B 1024 1 KiB used by this type, 4350 KiB unused memory. Operator Level>**AT+pUNLCK<cr>** OK

9.1.19 AT+pRMVOICE – Remove Voice

Remove Voice command Removes phrase files from the database.

Syntax:	**AT+pRMVOICE=str**
Parameters:	str=<name> An ASCII string of max. 10 characters (letters or digits).
Default value:	
Scope:	User level
Result code:	**OK** if the phrase exists and gets successfully deleted, otherwise **ERROR DELETING DATA IN DATABASE**
Example:	**AT+pRMVOICE=Voice1<cr>** OK

9.1.20 AT+pSCRIPTSTOR – Set or request Javascript storage values

The Script storage command is used to get or set values stored by JavaScripts.

Syntax:	**AT+pSCRIPTSTOR=str**
Parameters:	str=<name>:<content> name: The name the data should be stored as (length ≤ 10). content: The string to be saved. Leaving out the : or following it with a <CR> causes the data to be deleted.
Default value:	None.
Scope:	User level
Result code:	**OK** if the data is successfully stored, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pSCRIPTSTOR=weight:ok<cr>** OK
Note:	Values containing ASCII characters outside the range: [32-122] are not allowed to be configured through the Service interface and "{" , "}" cannot be used remotely. Length reported includes the zero termination at the end of strings.

Report a list of all stored values.

Syntax:	**AT+pSCRIPTSTOR?**
Scope:	User level
Result code:	Operator Level>**AT+pSCRIPTSTOR?<cr>** weight:ok 0 KiB used by this type, 4351 KiB unused memory.

9.1.21 AT+pINSSCRIPT – Insert or List Scripts

The Insert Script command adds a JavaScript to the database.

Syntax:	**AT+pINSSCRIPT=str**
Parameters:	An ASCII string of max. 10 characters (letters or digits). This string is the name for identifying the accompanying data as described in section 10.3, Uploading JavaScripts, sound files and images.
Default value:	None.
Scope:	User level
Result code:	**OK** if the script is successfully stored, otherwise **ERROR SAVING DATA IN DATABASE**
Example:	**--Binary file is transmitted and buffered—** Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSSCRIPT=script1<cr>** OK Operator Level>**AT+pUNLCK<cr>** OK **--Buffered binary data cleared--**
Note:	If a JavaScript with the same name exists on the system it will be replaced.

The format of the content will be JavaScript with custom objects for interfacing with the RTX337x system.

The name (parameter for the command) will be used by the system for identifying this JavaScript. The name will be used in configuration of devices referencing the JavaScript, and afterwards used by the device, when it determines that the JavaScript should be activated.

Report a list of all inserted JavaScript files.

Syntax:	**AT+pINSSCRIPT?**
Scope:	User level
Result code:	List of all Script files, and their size. Also prints a status of used and unused storage.
Example:	Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSSCRIPT?<cr>** A 20 B 1024 1 KiB used by this type, 4350 KiB unused memory. Operator Level>**AT+pUNLCK<cr>** OK

9.1.22 AT+pRMSCRIPT – Remove Script

Remove Script command removes JavaScript files from the database.

Syntax:	**AT+pRMSCRIPT=str**
Parameters:	str=<name> An ASCII string of max. 10 characters (letters or digits).
Default value:	
Scope:	User level
Result code:	**OK** if the script exist and gets successfully deleted, otherwise **ERROR DELETING DATA IN DATABASE**
Example:	**AT+pRMSCRIPT=Script1<cr>** OK

9.1.23 AT+pINSIMAGE – Insert or List Image

The Insert Image command adds image files to the database.

Syntax:	**AT+pINSIMAGE=str**
Parameters:	An ASCII string of max. 10 characters (letters or digits). This string is the name for identifying the accompanying binary data as described in section 10.3, Uploading JavaScripts, sound files and images.
Default value:	None.
Scope:	User level
Result code:	**OK** if the image is successfully stored, otherwise **ERROR SAVING DATA IN DATABASE**
Example:	**--Binary file is transmitted and buffered—** Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSIMAGE=image1<cr>** OK Operator Level>**AT+pUNLCK<cr>** OK **--Buffered binary data cleared--**
Note:	If an image file with the same name exists on the system it will be replaced.

The name (parameter for the command) will be used by the system for identifying the image in the binary data accompanying the configuration batch. The name will also be used in question trees for referencing the image.

Report a list of all inserted image files.

Syntax:	**AT+pINSIMAGE?**
Scope:	User level
Result code:	List of all image files, and their size. Also prints a status of used and unused storage.
Example:	**--Binary file is transmitted and buffered—** Operator Level>**AT+pLCK<cr>** OK Operator Level>**AT+pINSIMAGE?<cr>** Image1 3369 Image2 31424 33 KiB used by this type, 4319 KiB unused memory. Operator Level>**AT+pUNLCK<cr>** OK **--Buffered binary data cleared--**

9.1.24 AT+pRMIMAGE – Remove Image

Remove Image command Removes image files from the database.

Syntax:	**AT+pRMIMAGE=str**
Parameters:	str=<name> An ASCII string of max. 10 characters (letters or digits).
Default value:	
Scope:	User level
Result code:	**OK** if the image exists and gets successfully deleted, otherwise **ERROR DELETING DATA IN DATABASE**

Example:	**AT+pRMIMAGE=Image1<cr>** OK

9.1.25 AT+pINSFONT – Insert or List TrueType fonts

The insert Font command adds TrueType font file to the database.

Syntax:	**AT+pINSFONT=str**
Parameters:	str=<name> An ASCII string of max. 10 characters (letters or digits) including the extension ".ttf" which is required. This string is the name for identifying the accompanying binary data as described in section 10.3, Uploading JavaScripts, sound files, images and fonts.
Default value:	None.
Scope:	User level
Result code:	**OK** if the image is successfully stored, otherwise **ERROR SAVING DATA IN DATABASE**
Example:	**--Binary file is transmitted and buffered—** Operator Level>AT+pLCK<cr> OK Operator Level>AT+pINSFONT=Bold.ttf<cr> OK Operator Level>AT+pUNLCK<cr> OK **--Buffered binary data cleared--**
Note:	If a font file with the same name exists on the system it will be replaced. Inserted fonts has to be assigned by AT+pFONTASSIGN except Bold.ttf and Normal.ttf which is assigned by default. If any inserted file is assigned all text strings in JavaScript's and voice files are treated as UTF-8 instead of Latin-1 and also generated XML telegrams will be UTF-8 encoded. Any telegrams already queued will still have the old encoding. When changing from a RTX337x running without fonts to one running with fonts it should be ensured that scripts, text in phrase files, Log and scriptstorage area does not contain any iso-8859-1 specific character sequences as this will mean XML generated may contain non UTF8 sequences.

The name (parameter for the command) will be used by the system for identifying the font in the binary data accompanying the configuration batch.

Syntax:	**AT+pINSFONT?**
Scope:	User level
Result code:	List of all font files, and their sizes. Also prints a status of used and unused storage.
Example:	**--Binary file is transmitted and buffered—** Operator Level>AT+pLCK<cr> OK Operator Level>AT+pINSFONT?<cr> Bold.ttf 195956 Normal.ttf 198864 385 KiB used by this type, 2049 KiB unused memory. Operator Level>AT+pUNLCK<cr> OK **--Buffered binary data cleared--**

9.1.26 AT+pRMFONT – Remove Font

Remove Font command Removes font files from the database.

Syntax:	**AT+pRMFONT=str**
Parameters:	str=<name> An ASCII string of max. 10 characters (letters or digits). This string is the name of the font file to be deleted from the database.
Default value:	
Scope:	User level
Result code:	**OK** if the font file exists and gets successfully deleted, otherwise **ERROR DELETING DATA IN DATABASE**
Example:	Operator Level>**AT+pRMFONT=Bold.ttf<cr>** **OK**

9.1.27 AT+pFONTASSIGN – Assignment Of Truetype Fonts

Sets the assignment of TrueType fonts.

Syntax:	**AT+pFONTASSIGN=str** (Length of str < 50 characters)
Parameters:	str = A string of TrueType font filenames: <Normal font>,<Big font>,<Small font>,<Small slim font>. If value is undefined default values are used. Default values: Normal font = Bold.ttf Big font = Bold.ttf Small font = Bold.ttf Small slim font = Normal.ttf
Default value:	Empty string. Default values are used.
Scope:	Operator level
Result code:	**OK.**
Example:	Operator Level>**AT+pFONTASSIGN=Font,,Fon.ttf,Font <cr>** **OK**
Note:	If a not installed file is defined, the built-in font is used. (NB! No error message if a not installed file is defined). If one or more TrueType fonts are used, all text strings in JavaScript's and voice files are treated as UTF-8 instead of Latin-1 and also generated XML telegrams will be UTF-8 encoded. Any telegrams already queued will still have the old encoding. When changing from a RTX337x running without fonts to one running with fonts, it should be ensured that scripts, text in phrase files, log and scriptstorage area do not contain any iso-8859-1 specific character sequences. This would result in generated XML telegrams containing non UTF8 sequences.

Report the current setting. If no assignments exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pFONTASSIGN?**
Scope:	User level
Result code:	Font1,Font2,Fon.ttf,Font1 (example)
Example:	Operator Level>**AT+pFONTSIZE?<cr>** Font,,Fon.ttf,Font

9.1.28 AT+pFONTSIZE – Size of the TrueType fonts

Sets the size of the TrueType.

Syntax:	**AT+pFONTSIZE=str** (Length of str < 20 characters)
Parameters:	str = A string of TrueType font sizes: <Normal font>,<Big font>,<Small font>,<Small slim font>. If value is undefined or < 10 or > 50, default values are used. Default values: Normal font = 28 Big font = 36 Small font = 24 Small slim font = 24
Default value:	Empty string. Default value are used.
Scope:	Operator level
Result code:	**OK.**
Example:	Operator Level>**AT+pFONTSIZE=23,31,20,19<cr>** **OK**
Note:	This command has no effect on the built-in fonts which is used if no TrueType fonts are installed.

Report the current setting. If the font size does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pFONTSIZE?**
Scope:	User level
Result code:	23,31,20,19 (example)
Example:	Operator Level>**AT+pFONTSIZE?<cr>** 23,31,20,19

9.1.29 AT+pBACKGROUND –Set or List BACKGROUND images

Sets the names of the images to be used as background.

Syntax:	**AT+pBACKGROUND=str**
Parameters:	str=<name1>:<name2> Two ASCII names of max. 10 characters each (letters or digits). These strings are the names of the image files to be used as background images for all windows. The first file is used as basic background. The second file must be a copy of the basic background with, soft-buttons, menu highlight bares and arrows indicating scroll, added.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it names is valid, otherwise **ERROR - Names must be a maximum of 10 chars.**
Example:	Operator Level>**AT+pBACKGROUND=image1:image2<cr>** OK

Report the current background images.

Syntax:	**AT+pBACKGROUND?**
Scope:	User level
Result code:	Returns the current background image names.
Example:	Operator Level>**AT+pBACKGROUND?<cr>** Image1:image2

9.1.30 AT+pMODE – Set or List the current mode

Sets the operation mode.

<table>
<tr><td><b>Syntax:</b></td><td><b>AT+pMODE=str</b></td></tr>
<tr><td>Parameters:</td><td>The operating mode which to put the device in, must be one of FACTORY, PATIENT, NORMAL.

The only operating differences in the three modes are which JavaScript to run as default and the status of the RS232.
<table>
<tr><th>Mode</th><th>Running Script</th><th>Default RS232 status</th></tr>
<tr><td>Normal</td><td>Normal script.
(set by AT+pNORMALSCRIPT)</td><td>External device Interface.</td></tr>
<tr><td>Patient</td><td>Patient script.
(set by AT+pPATIENTSCRIPT)</td><td>External device Interface.</td></tr>
<tr><td>Factory</td><td>Factory script.
(set by AT+pFACTORYSCRIPT)</td><td>Technical Function Interface</td></tr>
</table>
It is possible to enter factory mode by pressing the keys: scroll down, No and volume down simultaneously for 5 sec. to ensure access to the Technical Function Interface.</td></tr>
<tr><td>Default value:</td><td>FACTORY</td></tr>
<tr><td>Scope:</td><td>Operator level</td></tr>
<tr><td>Result code:</td><td><b>OK</b> if it is a valid mode, otherwise <b>ERROR - Not a valid mode.</b></td></tr>
<tr><td>Example:</td><td>Operator Level><b>AT+pMODE=NORMAL&lt;cr&gt;</b>
OK</td></tr>
</table>

Report the current mode setting.

Syntax:	**AT+pMODE?**
Scope:	User level
Result code:	Returns the current mode.
Example:	Operator Level>**AT+pMODE?<cr>** NORMAL

9.1.31 AT+pFACTORYSCRIPT –Set or List FACTORY mode script name

Sets the name of the JavaScript to be by default activated in FACTORY mode.

Syntax:	**AT+pFACTORYSCRIPT=str**
Parameters:	The ASCII name of max. 10 characters (letters or digits). This string is the name of the JavaScript file to be activated when the device is in FACTORY mode.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it is a valid name, otherwise **ERROR - Name must be a maximum of 10 chars.**
Example:	Operator Level>**AT+pFACTORYSCRIPT=facscript<cr>** OK

Report the current Factory script.

Syntax:	**AT+pFACTORYSCRIPT?**
Scope:	User level
Result code:	Returns the current factory mode JavaScript name.
Example:	Operator Level>**AT+pFACTORYSCRIPT?<cr>** facscript

9.1.32 AT+pPATIENTSCRIPT – Set or List PATIENT mode script name

Sets the name of the JavaScript to be activated by default in PATIENT mode.

Syntax:	**AT+pPATIENTSCRIPT=str**
Parameters:	The ASCII name of max. 10 characters (letters or digits). This string is the name of the JavaScript file to be activated when the device is in PATIENT mode.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it is a valid name, otherwise **ERROR - Name must be a maximum of 10 chars.**
Example:	Operator Level>**AT+pPATIENTSCRIPT=patscript<cr>** OK

Report the current Patient script.

Syntax:	**AT+pPATIENTSCRIPT?**
Scope:	User level
Result code:	Returns the current PATIENT mode JavaScript name.
Example:	Operator Level>**AT+pPATIENTSCRIPT?<cr>** patscript

9.1.33 AT+pNORMALSCRIPT –Set or List NORMAL mode script name

Sets the name of the JavaScript to be activated by default in NORMAL mode.

Syntax:	**AT+pNORMALSCRIPT=str**
Parameters:	The ASCII name of max. 10 characters (letters or digits). This string is the name of the JavaScript file to be activated when the device is in NORMAL mode.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it is a valid name, otherwise **ERROR - Name must be a maximum of 10 chars.**
Example:	Operator Level>**AT+pNORMALSCRIPT=norscript<cr>** OK

Report the current Normal script.

Syntax:	**AT+pNORMALSCRIPT?**
Scope:	User level
Result code:	Returns the current NORMAL mode JavaScript name.
Example:	Operator Level>**AT+pNORMALSCRIPT?<cr>** norscript

9.1.34 AT+pTECHSCRIPT –Set or List Technical Menu script names

Sets one or more key combination and timeouts for activating JavaScripts. There can be as many name-keys-timeout pairs as allowed by the maximum length.

Syntax:	**AT+pTECHSCRIPT=str**
Parameters:	Str =name-keys-timeout:name-keys-timeout:… (maximum of 100 chars) ASCII names of JavaScript files to be activated with the specified key combination and timeout. The keys are specified as a decimal representation of the following bits (set to 1 for activating the button): Bit 0: Volume up button Bit 1: Volume down button Bit 2: Left selection button Bit 3: Select down Bit 4: Right selection button Bit 5: Select up Bit 6: Info button And the timeout is in seconds
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it is a valid name, otherwise **ERROR – String not accepted.**
Example:	Operator Level>**AT+pTECHSCRIPT=techscript-40-3:factreset-37-3<cr>** OK
Note:	Keys in the pairs (0,1), (2,3) and (4,5), are mutually exclusive, so they must not be used in combinations. The key combination 26 (1,3,4) is used for forcing the device into Factory mode, with a timeout of 5 seconds. If 2 JavaScripts have the same key combination the shortest timeout is used and will activate both events after that time (execution occurs in the order they are configured). If one key combination is included in another longer one only the matching one will be activated. JavaScripts activated in this way is always activated using the maximum priority, the activatorID "keyComb" and no parameters.

Report the current Technical script(s).

Syntax:	**AT+pTECHSCRIPT?**
Scope:	User level
Result code:	Returns the current Technical menu JavaScript configuration.
Example:	Operator Level>**AT+pTECHSCRIPT?<cr>** **techscript-40-3:factreset-37-3**

9.1.35 AT+pBOOTSCRIPT – Set or list Boot script name

Sets the name of the script to be activated at boot up.

Syntax:	**AT+pBOOTSCRIPT=str**
Parameters:	The ASCII name of max. 10 characters (letters or digits). This string is the name of the script file to be activated at boot up.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if it is a valid name, otherwise **ERROR - Name must be a maximum of 10 chars.**
Example:	Operator Level>**AT+pBOOTSCRIPT=bootscript<cr>** OK

Report the current Boot script.

Syntax:	**AT+pBOOTSCRIPT?**
Scope:	User level
Result code:	Returns the current boot up JavaScript configuration.
Example:	Operator Level> **AT+pBOOTSCRIPT?<cr>** bootscript

9.1.36 AT+pVOLUME –Set or List volume configuration

Sets the number of volume steps, their value and the default volume step.

<table>
<tr><td><b>Syntax:</b></td><td><b>AT+pVOLUME=str</b></td></tr>
<tr><td>Parameters:</td><td>A comma-separated list of decimal values following this syntax:

X_1,X_2,X_3,X_4…..X_n,Y
or
X_1,X_2,X_3,X_4…..X_n,Y:fname

The <i>X</i> values represent volume steps, i.e. if the list contains 5 X values the user will have 5 volume steps available. The list must contain between 1 and 10 steps. The value of each step must be in the range 0-25 where 0 is volume off and 25 is max volume.

The <i>Y</i> value is the default volume step. This is the step that will be selected after power-up. Please note that the list of steps is 0-based, i.e. the first volume step is step 0, the next is step 1 etc.

The <i>fname</i> is the name of the soundfile which is played when the volume buttons are pressed. The name must consist of letters and/or digits only. If no name is specified the volume feedback sound is disabled.

Examples:
<b>AT+pVOLUME=2,4,6,10,15,2</b>

This string will create 5 volume steps with the values 2, 4, 6, 10 and 15 and the default volume step is 2. After power-up the volume control will be set to 6. If the user decreases the volume it will become 4. If the user instead increases the volume, it will become 10.

<table>
<tr><td>Step</td><td>0</td><td>1</td><td>2
(default)</td><td>3</td><td>4</td></tr>
<tr><td>Value</td><td>2</td><td>4</td><td>6</td><td>10</td><td>15</td></tr>
</table>

<b>AT+pVOLUME=2,4,6,10,15,2:VolAdj</b>

This example will enable the volume feedback sound and play the sound file <i>VolAdj</i> when the volume buttons are pressed.

The list must be a maximum of 50 chars long.</td></tr>
<tr><td>Default value:</td><td>0,2,4,6,8,10,12,14,16,18,6:VolAdj</td></tr>
<tr><td>Scope:</td><td>User level</td></tr>
<tr><td>Result code:</td><td><b>OK</b> if it is a valid list, otherwise <b>ERROR – syntax error</b></td></tr>
<tr><td>Example:</td><td>User Level&gt;<b>AT+pVOLUME=2,4,6,10,15,2&lt;cr&gt;</b>
OK</td></tr>
</table>

Report the current volume setting.

Syntax:	**AT+pVOLUME?**
Scope:	User level
Result code:	Returns the current volume configuration
Example:	User Level>**AT+pVOLUME?<cr>** `2,4,6,10,15,2`

9.1.37 AT+pBACKLIGHT –Set or List backlight configuration

Sets the intensity level for dimmed and full intensity, as well as the power-up on-time and the fade in/fade out speed.

Syntax:	**AT+pBACKLIGHT=str**
Parameters:	A comma-separated list of decimal values following this syntax: $L_{normal}, L_{dimmed}, T_{powerup}, T_{fade}$ $L_{normal:}$ the intensity level used when normal intensity is selected. The value must be in the range 0-100. $L_{dimmed:}$ the intensity level used when dimmed intensity is selected. The value must be in the range 0-100. $T_{powerup:}$ The time, in 10 ms ticks, the display will stay lit after power-up. The intensity of the backlight in this period is set to 100% and is not configurable. The value must be in the range 0-65535. $T_{fade:}$ The time used to control fade in and fade out. During fading the intensity is adjusted every 10ms by T_{fade} steps. Thus a value of 2 causes a 100% fading to take 100%/2*10ms = 500ms. The value must be in the range 1-100.
Default value:	90,60,1000,2
Scope:	User level
Result code:	**OK** if it is a valid list, otherwise **ERROR – syntax error**
Example:	User Level>**AT+pBACKLIGHT=90,60,1000,2<cr>** OK This string will set normal and dimmed intensities to 90% and 60% respectively, power-up backlight time to 10 seconds and the fading time to half a second.

Report the current backlight configuration.

Syntax:	**AT+pBACKLIGHT?**
Scope:	User level
Result code:	Returns the current backlight configuration
Example:	User Level>**AT+pBACKLIGHT?<cr>** **90,60,1000,2**

9.1.38 AT+pBTLOCALNAME – Bluetooth Module Local Name

Sets the Bluetooth Module Local Name.

Syntax:	**AT+pBTLOCALNAME=str**
Parameters:	Str=Name The character string length must be in the range 2 - 15
Default value:	The default value is "RTX337x".
Scope:	Operator level
Result code:	**OK** if it names is valid, otherwise **Syntax error - Name must be 2 to 15 characters long.**
Example:	**AT+pBTLOCALNAME**=RTX337X<cr> OK

Report the current Bluetooth Module Local Name setting.

Syntax:	**AT+pBTLOCALNAME?**
Scope:	Operator level
Result code:	Returns the current Local Name setting
Example:	**AT+pBTLOCALNAME?<cr>** RTX337X

9.2 Commands for RTX337x Communication

Phone Dialing Commands

- **Direct connection**

Characteristics of phone lines can be different for different teleoperators. These variations are normally taken care of by a country code dependant initialization of the modem (an AT command). In the application this is handled by means of AT+p configuration commands in combination with JavaScript (Technical Menu) and user interaction. There is one exception, which can be automatically determined relatively easy by JavaScript:

*DTMF or pulse dialing.

- **PBX**

There are many different PBX's with different characteristics. These can be handled by means of AT+p configuration commands in combination with JavaScript and user interaction. It will be very complicated, because of the big variety of PBX characteristics, to automate any of this.

- **Voice mail**

Voice mail can be handled by not waiting for a dial tone, when dialling the Internet Service Provider (ISP). This can be handled by means of AT+p configuration commands in combination with JavaScript and user interaction. This cannot be automated.

- **Call waiting**

To enable/disable "call waiting" is different for different tele operators. Some supports turning off "call waiting" for the duration of a call. Others requires first turning off "call waiting", then dial and after the call is finished "call waiting" must be enabled again. The code for doing this varies between teleoperators. This can be handled by means of AT+p configuration commands in combination with JavaScript and user interaction. This cannot be automated.

As described above, most of the adaption to different phone lines cannot be automated or is very difficult to automate. To ease the adaption to different phone lines a semi automatical procedure is used. To achieve this we use configuration parameters in the database.

9.2.1 AT+pISP – ISP Phone Number

Sets the phone number to the ISP (Internet Service Providers) dial-in service.

Syntax:	**AT+pISP=str** (Length of str < 50 characters)
Parameters:	str = Telephone number string. See description of the ATD command for the modem in "Silicon Laboratories SI2415". This is the number used when the Gateway tries to connect to the Internet. The string is used as the last part of the main dial command for the modem. The main dial command consist of the MPRED1 (see 9.2.10) string followed by the MPRED2 (see 9.2.12) string followed by the MPRED3 (see 9.2.14) string and finally followed by the ISP string. "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital. For GPRS connect on RTX3371: *<gprs_sc>***1# <gprs_sc> - GPRS Service Code, a digit string (value 99) which identifies a request to use the GPRS
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 50 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pISP=96682900<cr>** OK *Dials the phone number "96682900"* **AT+pISP=*99***1#<cr>** *OK* *Connects via GPRS (RTX3371 only)*

Report the current value of the ISP's phone number. If the number does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pISP?**
Scope:	User level
Result code:	12345678 (example)
Example:	**AT+pISP?<cr>** 96682900 *Retrieves the current phone number*

9.2.2 AT+pISP_BACKUP – ISP Backup Phone Number

Sets the backup phone number to the ISP dial-in service. This may be a toll-free number.

Syntax:	**AT+pISP_BACKUP=str** (Length of str < 50 characters)
Parameters:	str = Telephone number string. See description of the ATD command for the modem. This is the number used when the RTX337x tries to connect to the internet and ISP or GPRS connection failed with the primary number, username and password. The string is used as the last part of the main dial command for the modem. The main dial command consist of the MPRED1 (see section 9.2.10, AT+pMPRED1) string followed by the MPRED2 (see section 9.2.12, AT+pMPRED2) string followed by the MPRED3 (see section 9.2.14, AT+pMPRED3) string and finally followed by the ISP value. "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 50 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pISP_BACKUP=18005551212<cr>** OK

Report the current value of the backup ISP phone number. If the number does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pISP_BACKUP?**
Scope:	User level
Result code:	18005551212 (example)
Example:	**AT+pISP_BACKUP?<cr>** 18005551212

9.2.3 AT+pISP_VC – ISP Phone Number Vector

Sets a string vector with a number of ISP or ISP_BACKUP phone numbers.

Syntax:	**AT+pISP_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of ISP or ISP_BACKUP phone numbers separated with ":". Entries in this vector can be used for initialising ISP or ISP_BACKUP from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pISP_VC=517653:634978:934658 <cr>** OK

Report the current value of the ISP phone number vector. If the vector does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pISP_VC?**
Scope:	User level
Result code:	517653:634978:934658 (example)
Example:	**AT+pISP_VC?<cr>** 517653:634978:934658

9.2.4 AT+pMBEG1 – Init String for Modem

Sets an initialization string for the "Silicon Laboratories SI2415" modem.

Syntax:	**AT+pMBEG1=str** (Length of str < 50 characters)
Parameters:	str = string with init string (excluding "AT") for the modem. The initialization command will be issued as the first configuration command before dialling. It can be a combined command with more than one AT command. "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMBEG1=+GCI=B5<cr>** OK *Modem setting for North America* **AT+pMBEG1=+GCI=42<cr>** OK *Modem setting for Europe*

Report the current value of the modem string. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMBEG1?**
Scope:	User level
Result code:	+GCI=B5 (example)
Example:	**AT+pMBEG1?<cr>** +GCI=B5 *Return current init string*

9.2.5 AT+pMBEG1_VC – Init string Vector

Sets a string vector with a number of MBEG1 initialization strings.

Syntax:	**AT+pMBEG1_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MBEG1 initialization strings separated with ":". Entries in this vector can be used for initialising MBEG1 from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMBEG1_VC=+GCI=31:+GCI=B5 <cr>** OK

Report the current value of the MBEG1 initialization string vector. If the vector does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMBEG1_VC?**
Scope:	User level
Result code:	+GCI=31:+GCI=B5 (example)
Example:	**AT+pMBEG1_VC?<cr>** **=+GCI=31:+GCI=B5**

9.2.6 AT+pMBEG2 – Init String for Modem

Sets an initialization string for the "Silicon Laboratories SI2415" modem.

Syntax:	**AT+pMBEG2=str** (Length of str < 50 characters)
Parameters:	str = string with init string (excluding "AT") for the modem. The initialization command will be issued as the second configuration command before dialling. It can be a combined command with more than one AT command. "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMBEG2=X3<cr>** OK *Modem expecting no dial tone*
Note:	Be aware that modem configuration changes done with this command are after the RTX337x internal configuration. This means that it is possible to overrule any configuration of the modem made by the firmware. Special care should be taken if changes are made to register U76 and U77, which control the Line intrusion detection, since the line intrusion detection might not work as intended causing the phone to be blocked for other purposes, e.g. emergency calls.

Report the current value of the modem string. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMBEG2?**
Scope:	User level
Result code:	X3 (example)
Example:	**AT+pMBEG2?<cr>** X3 *Return current init string*

9.2.7 AT+pMBEG2_VC – Init string Vector

Sets a string vector with a number of MBEG2 initialization strings.

Syntax:	**AT+pMBEG2_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MBEG2 initialization strings separated with ":". Entries in this vector can be used for initialising MBEG2 from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMBEG2_VC=X3:X1 <cr>** OK

Report the current value of the MBEG2 initialization string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMBEG2_VC?**
Scope:	User level
Result code:	X3:X1(example)
Example:	**AT+pMBEG2_VC?<cr>** **X3:X1**

9.2.8 AT+pMCDBEG – Control Dial String for Modem

This is a string with an AT control dial command (ATD) for the modem.

Syntax:	**AT+pMCDBEG=str** (Length of str < 50 characters)
Parameters:	str = string with a control dial command (excluding "ATD") for the modem. . The command will be issued before the "main" dial. It could be for turning off "call waiting". The string can include %s placeholders (max. two). The first placeholder (if any) is substituted with the MPRED1 (see later) string and the second placeholder (if any) is substituted with the MPRED2 (see later) string. This can be used for the control dial command, where MPRED1 distinguishes between DTMF and PULSE dial (ex. T), and MPRED2 is the PBX dial command (ex. 9W). "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMCDBEG=%s*43#<cr>** OK *Turn off "call waiting". Use tone or pulse dial as prescribed by MPRED1.*

Report the current value of the modem string. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMCDBEG?**
Scope:	User level
Result code:	*43# (example)
Example:	**AT+pMCDBEG?<cr>** %s*43# *Return current control dial string*

9.2.9 AT+pMCDBEG_VC – Control Dial String Vector

Sets a string vector with a number of MCDBEG Control Dial strings.

Syntax:	**AT+pMCDBEG_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MCDBEG initialization strings separated with ":". Entries in this vector can be used for initialising MCDBEG from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMCDBEG_VC=*43#:*49# <cr>** OK

Report the current value of the MCDBEG control dial string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMCDBEG_VC?**
Scope:	User level
Result code:	*43#:*49#(example)
Example:	**AT+pMCDBEG_VC?<cr>** ***43#:*49#**

9.2.10 AT+pMPRED1 – First Pre Dial String for Modem

Sets the first pre dial string for the modem dial command.

Syntax:	**AT+pMPRED1=str** (Length of str < 50 characters)
Parameters:	str = string with first pre dial string for the modem. This is the first part of the main dial (for calling the ISP) command (ATD) to the modem. It could be for distinguishing between DTMF and PULSE dial (ex. T). "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED1=T<cr>** OK *Modem dial with tone dial*

Report the current value of the first pre dial string for the modem. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMPRED1?**
Scope:	User level
Result code:	T (example)
Example:	**AT+pMPRED1?<cr>** T *Return current string*

9.2.11 AT+pMPRED1_VC – Pre Dial String Vector

Sets a string vector with a number of MPRED1 pre dial strings.

Syntax:	**AT+pMPRED1_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MPRED1 pre dialstrings separated with ":". Entries in this vector can be used for initialising MPRED1 from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED1_VC=T:P <cr>** OK

Report the current value of the MPRED1 pre dial string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMPRED1_VC?**
Scope:	User level
Result code:	T:P(example)
Example:	**AT+pMPRED1_VC?<cr>** **T:P**

9.2.12 AT+pMPRED2 – Second Pre Dial String for Modem

Sets the second pre dial string for the modem dial command.

Syntax:	**AT+pMPRED2=str** (Length of str < 50 characters)
Parameters:	str = string with second pre dial string for the modem. This is the second part of the main dial (for calling the ISP) command (ATD) to the modem. It could be PBX dial (ex. 9W). "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED2=9W<cr>** OK *Modem dial PBX with 9 and wait for dial tone*

Report the current value of the second pre dial string for the modem. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMPRED2?**
Scope:	User level
Result code:	9W (example)
Example:	**AT+pMPRED2?<cr>** 9W *Return current string*

9.2.13 AT+pMPRED2_VC – Pre Dial String Vector

Sets a string vector with a number of MPRED2 pre dial strings.

Syntax:	**AT+pMPRED2_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MPRED2 pre dialstrings separated with ":". Entries in this vector can be used for initialising MPRED2 from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED2_VC=:9W <cr>** OK

Report the current value of the MPRED2 pre dial string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMPRED2_VC?**
Scope:	User level
Result code:	:9W(example)
Example:	**AT+pMPRED2_VC?<cr>** :9W

9.2.14 AT+pMPRED3 – Third Pre Dial String for Modem

Sets the third pre dial string for the modem dial command.

Syntax:	**AT+pMPRED3=str** (Length of str < 50 characters)
Parameters:	str = string with third pre dial string for the modem. This is the third part of the main dial (for calling the ISP) command (ATD) to the modem. It could be turn off "call waiting" for the duration of the call (ex. *70,). "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED3=*70,<cr>** OK *Turn off "call waiting" for the duration of the call*

Report the current value of the third pre dial string for the modem. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMPRED3?**
Scope:	User level
Result code:	*70, (example)
Example:	**AT+pMPRED3?<cr>** ***70,** *Return current string*

9.2.15 AT+pMPRED3_VC – Pre Dial String Vector

Sets a string vector with a number of MPRED3 pre dial strings.

Syntax:	**AT+pMPRED3_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MPRED3 pre dialstrings separated with ":". Entries in this vector can be used for initialising MPRED3 from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMPRED3_VC=*70,:*76, <cr>** OK

Report the current value of the MPRED3 pre dial string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMPRED3_VC?**
Scope:	User level
Result code:	*70,:*76,(example)
Example:	**AT+pMPRED3_VC?<cr>** ***70,:*76,**

9.2.16 AT+pMEND – Clean Up String for Modem

Sets a clean up string for the modem.

Syntax:	**AT+pMEND=str** (Length of str < 50 characters)
Parameters:	str = string with clean up AT command (excluding "AT") for the modem. The command will be issued after the termination of the call. "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMEND=<cr>** OK *No modem clean up command*

Report the current value of the modem clean up string. If the string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMEND?**
Scope:	User level
Result code:	(example)
Example:	**AT+pMEND?<cr>** *Return current cleanup string*

9.2.17 AT+pMEND_VC – Clean Up String Vector

Sets a string vector with a number of MEND clean up strings.

Syntax:	**AT+pMEND_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MEND clean up strings separated with ":". Entries in this vector can be used for initialising MEND from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMEND_VC= <cr>** OK

Report the current value of the MEND cleanup string vector. If the vector does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMEND_VC?**
Scope:	User level
Result code:	(example)
Example:	**AT+pMEND_VC?<cr>**

9.2.18 AT+pMCDEND – Control Dial String for Modem

This is a string with an AT control dial command (ATD) for the modem.

Syntax:	**AT+pMCDEND=str** (Length of str < 50 characters)
Parameters:	str = string with the control dial string (excluding "ATD") for the modem. The command will be issued after the termination of the main dial. It could be for turning off "call waiting". The string can include %s placeholders (max. two). The first placeholder is substituted with the MPRED1 string and the second placeholder is substituted with the MPRED2 string. This can be used for the control dial command where MPRED1 distinguishes between DTMF and PULSE dial (ex. T), and MPRED2 is the PBX dial command (ex. 9W). "{" and "}" characters cannot be used when the command is used remotely. Letters must be capital.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMCDEND=%s#43#<cr>** OK *Modem expecting no dial tone*

Report the current value of the modem init string. If the init string does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pMCDEND?**
Scope:	User level
Result code:	%s#43# (example)
Example:	**AT+pMCDEND?<cr>** %s#43# *Return current init string*

9.2.19 AT+pMCDEND_VC – Control Dial String Vector

Sets a string vector with a number of MCDEND Control Dial strings.

Syntax:	**AT+pMCDEND_VC=str** (Length of str < 510 characters)
Parameters:	str = A string vector with a number of MCDEND control dial strings separated with ":". Entries in this vector can be used for initialising MCDEND from a JavaScript.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 510 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pMCDEND_VC=#43#:#49# <cr>** OK

Report the current value of the MCDEND control dial string vector. If the vector does not exist (empty string), only <CRLF> is returned

Syntax:	**AT+pMCDEND_VC?**
Scope:	User level
Result code:	**#43#:**#49#(example)
Example:	**AT+pMCDEND_VC?<cr>** **#43#:#49#**

9.2.20 AT+pPINCODE - PIN code in the RTX3371

Sets the PIN code that is sent from the RTX337x to the SIM card (to access the SIM Card). The PIN code for the SIM Card cannot be changed from here. (NB! Only relevant for RTX3371). See section 6.4 for more information about SIM Card and PIN code.

Syntax:	**AT+pPINCODE =str** (Length of str < 9 characters)
Parameters:	str = PIN code string. If the SIM card is PIN code protected, this value is used as access code for the SIM card.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 9 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pPINCODE=1234<cr>** OK

Report the current value of PIN code in the RTX3371. If the string is empty, only <CRLF> is returned

Syntax:	**AT+pPINCODE?**
Scope:	Operator level
Result code:	1234 (example)
Example:	**AT+pPINCODE?<cr>** 1234

9.2.21 AT+pGPRSAPN - GPRS Access Point Name

Sets the GPRS Access Point Name. This information must be provided by your GPRS/GSM network provider. (NB! Only relevant for RTX3371)

Syntax:	**AT+pGPRSAPN =str** (Length of str < 50 characters)
Parameters:	str = GPRS Access Point Name string.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if length of string < 50 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pGPRSAPN=internet<cr>** OK

Report the current value GPRS Access Point Name. . If the string is empty, only <CRLF> is returned

Syntax:	**AT+pGPRSAPN?**
Scope:	User level
Result code:	internet (example)
Example:	**AT+pGPRSAPN?<cr>** Internet

9.2.22 AT+pGPRSIP - IP address applicable to the GPRS PDP

Sets the GPRS IP address. This information must be provided by your GPRS/GSM network provider. (NB! Only relevant for RTX3371)

Syntax:	**AT+pGPRSIP=str** (Length of str < 20 characters)
Parameters:	str = GPRS IP address string.
Default value:	0.0.0.0 (dynamically assigned by the ISP)
Scope:	User level
Result code:	**OK** if length of string < 20 characters, otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pGPRSIP=167.35.46.3<cr>** OK

Report the current value of GPRS IP address. If the string is empty, only <CRLF> is returned.

Syntax:	**AT+pGPRSIP?**
Scope:	User level
Result code:	167.35.46.3 (example)
Example:	**AT+pGPRSIP?<cr>** 167.35.46.3

9.2.23 AT+pGPRSHEADCOMP - GPRS PDP header compression

Sets the GPRS PDP header compression. This information must be provided by your GPRS/GSM network provider. (NB! Only relevant for RTX3371)

Syntax:	**AT+pGPRSHEADCOMP=str** Length of str < 2 characters

Parameters:	str = 1 ⇔ GPRS PDP header compression: **ON**.
	str = 0 ⇔ GPRS PDP header compression: **OFF**.

Default value:	0
Scope:	User level
Result code:	**OK** if legal value Otherwise: **Syntax Error** - Value must be 0 or 1.
Example:	**AT+pGPRSHEADCOMP=1 <cr>** OK

Report the current value of GPRS PDP header compression.

Syntax:	**AT+pGPRSHEADCOMP?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pGPRSHEADCOMP?<cr>** 1

9.2.24 AT+pGPRSDATACOMP - GPRS PDP Data compression

Sets the GPRS PDP Data compression. This information must be provided by your GPRS/GSM network provider. (NB! Only relevant for RTX3371)

Syntax:	**AT+pGPRSDATACOMP=str** Length of str < 2 characters

Parameters:	str = 1 ⇔ GPRS PDP Data compression: **ON**.
	str = 0 ⇔ GPRS PDP Data compression: **OFF**.

Default value:	0
Scope:	User level
Result code:	**OK** if legal value Otherwise: **Syntax Error** - Value must be 0 or 1.
Example:	**AT+pGPRSDATACOMP=1 <cr>** OK

Report the current value of GPRS PDP Data compression.

Syntax:	**AT+pGPRSDATACOMP?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pGPRSDATACOMP?<cr>** 1

9.2.25 AT+pSIGNAL – GSM Signal Strength

Report the GSM Signal Strength. (NB! Only relevant for RTX3371)

Syntax:	**AT+pSIGNAL?**
Scope:	User level
Result code:	Signal Quality, Bit error rate, GPRS service state Signal Quality 0 - (-113) dBm or less 1 - (-111) dBm 2..30 - (-109)dBm..(-53)dBm / 2 dBm per step 31 - (-51)dBm or greater 99 - not known or not detectable Bit error rate (in percent) 0 = less than 0.2% 1 = 0.2% to 0.4% 2 = 0.4% to 0.8% 3 = 0.8% to 1.6% 4 = 1.6% to 3.2% 5 = 3.2% to 6.4% 6 = 6.4% to 12.8% 7 = more than 12.8% 99 = not known or not detectable GPRS service state 0 = detached 1 = attached
Example:	**AT+pSIGNAL? <cr>** 25,0,1

9.2.26 AT+pIMEI – GSM Module Identification

Report the GSM IMEI. (NB! Only relevant for RTX3371)

Syntax:	**AT+pIMEI?**
Scope:	User level
Result code:	15 digit number
Example:	**AT+pIMEI? <cr>** 123456789012345

Internet Connection Commands

9.2.27 AT+pWS – Web Server Address

Sets the address of the web server the RTX337x connects to, in order to transmit its collected data. For security reasons this command requires "Operator level" authorization for access.

Syntax:	**AT+pWS=str** (Length of str < 256 characters)
Parameters:	str = address. Either a host name or a numeric IP address format is valid. The string can be of the form IP:PORT or example.com:PORT.
Default value:	Empty string.
Scope:	Operator level
Result code:	**OK** if legal address string. Otherwise **Syntax Error. Not a FQHN (Fully Qualified Host Name).**
Example:	**AT+pWS=80.160.225.46<cr>** OK **AT+pWS=example.com:8081<cr>** OK

Report the current value of the web server address. If the address does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pWS?**
Scope:	User level
Result code:	80.160.225.57 (example).
Example:	**AT+pWS?<cr>** 80.160.225.46

9.2.28 AT+pWSPATH – Web Server Path

Sets the path to the web service on the web server.

Syntax:	**AT+pWSPATH=str** (Length of str < 256 characters)
Parameters:	str = path. The string must be a valid URL, so special characters must be escaped.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if legal path string. Otherwise **ERROR SAVING DATA IN DATABASE.**
Example:	**AT+pWSPATH=HCareSQL/Gw33xxSQL.asmx<cr>** OK

Report the current value of the web server Path. If the path does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pWSPATH?**
Scope:	User level
Result code:	HCareSQL/Gw33xxSQL.asmx (example)
Example:	**AT+pWSPATH?<cr>** HCareSQL/Gw33xxSQL.asmx

9.2.29 AT+pUSR – ISP or GPRS User Name

Sets the user name used for login to the ISP (Internet Service Provider) or GPRS.

Syntax:	**AT+pUSR=str** (Length of str < 50 characters)
Parameters:	str = string with user name. The user name is used by PPP for login to the ISP or GPRS. "{" and "}" characters cannot be used when the command is used remotely.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if legal user name string. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pUSR=username<cr>** OK

Report the current value of the user name. If the user name does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pUSR?**
Scope:	User level
Result code:	120000567483 (example)
Example:	**AT+pUSR?<cr>** **Username**

9.2.30 AT+pUSR_BACKUP – ISP or GPRS Backup User Name

Sets the backup user name used for login to the ISP (Internet Service Provider) or GPRS.

Syntax:	**AT+pUSR_BACKUP=str** (Length of str < 50 characters)
Parameters:	str = string with user name. The backup user name is used by PPP for login to the ISP or GPRS when the Gateway tries to connect to the internet with ISP backup phone number because ISP or GPRS connection failed with the primary number, username and password. "{" and "}" characters cannot be used when the command is used remotely.
Default value:	Empty string. (If empty primary user name is used)
Scope:	User level
Result code:	**OK** if legal user name string. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pUSR_BACKUP=username<cr>** OK

Report the current value of the backup user name. If the user name does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pUSR_BACKUP?**
Scope:	User level
Result code:	120000567483 (example)
Example:	**AT+pUSR_BACKUP?<cr>** Username

9.2.31 AT+pPSW – ISP or GPRS User Password

Sets the password used for login to the ISP (Internet Service Provider) or GPRS.

Syntax:	**AT+pPSW=str** (Length of str < 50 characters)
Parameters:	str = string with password. The password is used by PPP for login to the ISP or GPRS. "{" and "}" characters cannot be used when the command is used remotely.
Default value:	Empty string.
Scope:	User level
Result code:	**OK** if legal password string. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pPSW=password<cr>** OK

Report the current value of password. If the password does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pPSW?**
Scope:	User level
Result code:	c12USG5s (example)
Example:	**AT+pPSW?<cr>** **c12USG5s**

9.2.32 AT+pPSW_BACKUP – ISP or GPRS Backup User Password

Sets the backup password used for login to the ISP (Internet Service Provider) or GPRS.

Syntax:	**AT+pPSW_BACKUP=str** (Length of str < 50 characters)
Parameters:	str = string with password. The backup password is used by PPP for login to the ISP or GPRS when the Gateway tries to connect to the internet with ISP backup phone number because ISP or GPRS connection failed with the primary number, username and password. "{" and "}" characters cannot be used when the command is used remotely.
Default value:	Empty string. (If empty primary password is used)
Scope:	User level
Result code:	**OK** if legal password string. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pPSW_BACKUP=password<cr>** OK

Report the current value of backup password. If the password does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pPSW_BACKUP?**
Scope:	User level
Result code:	c12USG5s (example)
Example:	**AT+pPSW_BACKUP?<cr>** **c12USG5s**

9.2.33 AT+pDNS – DNS Server IP Address

Sets the numeric IP address of the DNS servers to be used. This must be set if the system is configured with a hostname for the host server. The DNS servers will be tried in the order specified and the first fulfilled request will be used.

Syntax:	**AT+pDNS=str** (Length of str < 50 characters)
Parameters:	str = A string vector of IP addresses, individual entries are separated with ":". Only a string with legal numeric IP address format is valid. Or empty string if no DNS server is used.
Default value:	Empty string. No DNS lookup is performed.
Scope:	Operator level
Result code:	**OK** if legal IP address string. Otherwise: **Syntax Error. Not an IP address.**
Example:	**AT+pDNS=80.160.225.46:80.160.224.47<cr**> OK

Report the current value of the DNS server IP address. If the proxy address does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pDNS?**
Scope:	User level
Result code:	80.160.225.46:80.160.224.47 (example)
Example:	**AT+pDNS?<cr>** **80.160.225.46:80.160.224.47**

9.2.34 AT+pPRXY – Proxy Address

Sets the address of a proxy server. This address is used in cases where the Internet server is behind a web proxy. The address should only be set when a web proxy server must be used.

Syntax:	**AT+pPRXY=str** (Length of str < 256 characters)
Parameters:	str = address. Either a host name or a numeric IP address format is valid. The string can be of the form IP:PORT or example.com:PORT.
Default value:	Empty string. No web proxy is used.
Scope:	User level
Result code:	**OK** if legal address string. Otherwise: **Syntax Error. Not a FQHN (Fully Qualified Host Name)**
Example:	**AT+pPRXY=80.160.225.46<cr>** OK **AT+pPRXY=80.160.225.46:8081<cr>** OK

Report the current value of the web proxy address. If the proxy address does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pPRXY?**
Scope:	User level
Result code:	80.160.225.46 (example)
Example:	**AT+pPRXY?<cr>** 80.160.225.46

9.2.35 AT+pGZIPCOMP - GZIP compression

Selects GZIP compression for data send to the server.

Syntax:	**AT+pGZIPCOMP=str**
Parameters:	str = <level> <level> = 0 to 9 0: GZIP compression: OFF. 1: Low compression (Lower CPU load). : : 9: High compression (Higher CPU load).
Default value:	0
Scope:	User level
Result code:	**OK** if legal value Otherwise: **Syntax Error. Character must be 0 to 9.**
Example:	**AT+pGZIPCOMP=6 <cr>** OK

Report the current value of GZIP compression.

Syntax:	**AT+pGZIPCOMP?**
Scope:	User level
Result code:	6 (example)
Example:	**AT+pGZIPCOMP?<cr>** 6

9.2.36 AT+pFALLBACK – Settings Fallback

Enables or disables fallback in case of error when contacting the web server.

Syntax:	**AT+pFALLBACK=str** Length of str < 2 characters
Parameters:	str = 1 ⇔ use fallback. Str = 0 ⇔ do not use fallback.
Default value:	0. Do not use fallback.
Scope:	User level
Result code:	**OK** if str is 0 or 1. Otherwise: **Syntax Error**. **Character must be 0 or 1.**
Example:	**AT+pFALLBACK=0<cr>** OK *This command turns OFF the use of fallback*

Report the current value of fallback.

Syntax:	**AT+pFALLBACK?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pFALLBACK?<cr>** **1** *This command reports the current fallback setting*

9.2.37 AT+pMTUSIZE – MTU Size

Sets the MTU size.

Syntax:	**AT+pMTUSIZE=str** Length of str < 5 characters
Parameters:	str = MTU size string. The smallest MTU size allowed is 128.
Default value:	232.
Scope:	User level
Result code:	**OK** if length of string < 5 characters and value is legal, otherwise **Syntax Error. Not a legal value.**
Example:	**AT+pMTUSIZE=1500<cr>** OK

Report the current value of fallback.

Syntax:	**AT+ pMTUSIZE?**
Scope:	Operator level
Result code:	1500 (example)
Example:	**AT+pMTUSIZE?<cr>** 1500

9.2.38 AT+pFORCEMTOMREQ – Force MTOM Request

Force the server to send binary data as MTOM attachments by sending a small dummy MTOM attachment with each telegram to the server.

Syntax:	**AT+pFORCEMTOMREQ=str**

Parameters:	str = 1 ⇔ Force MTOM Request: **ON**.
	str = 0 ⇔ Force MTOM Request: **OFF**.

Default value:	1
Scope:	User level
Result code:	**OK** if legal value Otherwise: **Syntax Error** - Value must be 0 or 1.
Example:	**AT+pFORCEMTOMREQ=1 <cr>** OK

Report the current value of fallback.

Syntax:	**AT+ pFORCEMTOMREQ?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pFORCEMTOMREQ?<cr>** 1

Bluetooth Connection Commands

9.2.39 AT+pCOD – Class of Device

Sets the Class of Device in the Bluetooth module. This setting controls how the Bluetooth module responds to inquiries from other Bluetooth devices.

Syntax:	**AT+pCOD=str** (Length of str < 8 characters)
Parameters:	str = CoD. The hex value of the CoD.
Default value:	520204
Scope:	Operator level
Result code:	**OK** if CoD is in legal range. Otherwise **Syntax Error. Input must be in the range 0x000000 to 0xFFFFFF**
Example:	**AT+pCOD=520204<cr>** OK *This command sets the CoD to 0x520204*

Report the current value of the CoD.

Syntax:	**AT+pCoD?**
Scope:	User level
Result code:	520204 (example)
Example:	**AT+pCOD?<cr>** **520204** *This command reports the current CoD setting*

9.3 Security Commands

9.3.1 AT+pSAUT – SSL Server Authentication

Sets (or clears) SSL server authentication. SSL authentication can only be enabled when the relevant certificates are present (in this case CA certificate).
If the web server address (see 9.2.27, AT+pWS, for details) does not specify a PORT the following defaults are used:

If SSL server authentication is OFF, PORT 80 is used.
If SSL server authentication is ON, PORT 443 is used.

Syntax:	**AT+pSAUT=str** Length of str < 2 characters

Parameters:	str = 1 ⇔ use SSL server authentication.
	str = 0 ⇔ do not use SSL server authentication. This also disables SSL client authentication.

Default value:	0. Do not use SSL server authentication.
Scope:	Operator level
Result code:	**OK** if str is 0 or 1. Otherwise: **Syntax Error**. **Character must be 0 or 1.**
Example:	**AT+pSAUT=0<cr>** OK *This command turns OFF the SSL server authentication* **AT+pSAUT=1<cr>** OK *This command turns ON the SSL server authentication*

Report the current value of the SSL server authentication.

Syntax:	**AT+pSAUT?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pSAUT?<cr>** **1** *This command displays the current server authentication setting*

9.3.2 AT+pCAUT – SSL Client Authentication

Sets (or clears) SSL client authentication. SSL authentication can only be done when the relevant certificates are present (in this case the client certificate and the private key).

Syntax:	**AT+pCAUT=str** Length of str < 2 characters

Parameters:	str = 1 ⇔ use SSL client authentication.
	str = 0 ⇔ do not use SSL client authentication.

Default value:	0. Do not use SSL client authentication.
Scope:	Operator level
Result code:	**OK** if str is 0 or 1. Otherwise: **Syntax Error**. **Character must be 0 or 1.**
Example:	**AT+pCAUT=0<cr>** OK *This command turns OFF the SSL client authentication* **AT+pCAUT=1<cr>** OK *This command turns ON the SSL client authentication*

Note:	Client authentication is active (SAUT=0) only if server authentication is also enabled ().

Report the current value of the SSL client authentication.

Syntax:	**AT+pCAUT?**
Scope:	User level
Result code:	1 (example)
Example:	**AT+pCAUT?<cr>** **1** *This command displays the current client authentication setting*

9.3.3 AT+pCACRT – SSL CA Certificate

Sets the CA (Certification Authority) certificate. Used by SSL server authentication. The command must be terminated with a <CR> character. Line Feed characters <LF> are accepted in the command.

Syntax:	**AT+pCACRT=str** Length of str < 8192 characters
Parameters:	Str = CA certificate, in PEM format.
Default value:	Empty string.
Scope:	Operator level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pCACRT=** **-----BEGIN CERTIFICATE-----** **MIIDFTCCAn6gAwIBAgIBADANBgkqhkiG9w0BAQQFADBrMQswCQYDGJSzEQ** **etc...** **-----END CERTIFICATE-----<cr>** OK

Report the current value of the CA certificate. If the certificate does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pCACRT?**
Scope:	Operator level
Result code:	(The certificate in PEM format)
Example:	**AT+pCACRT?<cr>** **-----BEGIN CERTIFICATE-----** **MIIDFTCCAn6gAwIBAgIBADANBgkqhkiG9w0BAQQFADBrMQswCQYDESzEQ** **etc...** **-----END CERTIFICATE-----**

9.3.4 AT+pCCRT – SSL Client Certificate

Sets the client certificate, including private key. Used by SSL client authentication. The command must be terminated with a <CR> character. Line Feed characters <LF> are accepted in the command.

Syntax:	**AT+pCCRT=str** Length of str < 4096 characters
Parameters:	Str = client certificate, including private key, in PEM format.
Default value:	Empty string.
Scope:	Operator level
Result code:	**OK** if accepted. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pCCRT=**

	-----BEGIN RSA PRIVATE KEY----- **Proc-Type: 4,ENCRYPTED** **DEK-Info: DES-EHE3-CBC,407A15EC9944B9BE** **TjbYmBlxlrz4SHMKbvBGdT8GDBzXu/v7P3z2AQldbfFJ3Q+lsfPVb7UjvAJ etc...** **-----END RSA PRIVATE KEY-----** **-----BEGIN CERTIFICATE-----** **MIICOzCCAaQCAQIwDQYJKoZIhvcNAQEEBQAwazELMAkGA1UEBhMCRE sgNVetc...** **-----END CERTIFICATE-----<cr>** OK
Note:	The password for the private key must be "password".

Report the current value of the client certificate. If the certificate does not exist (empty string), only <CRLF> is returned.

Syntax:	**AT+pCCRT?**
Scope:	Operator level
Result code:	(The certificate in PEM format)
Example:	**AT+pCCRT?<cr>** **-----BEGIN RSA PRIVATE KEY-----** **Proc-Type: 4,ENCRYPTED** **DEK-Info: DES-EHE3-CBC,407A15EC9944B9BE** **TjbYmBlxlrz4SHMKbvBGdT8GDBzXu/v7P3z2AQldbfFJ3Q+lsaVb7UjvAJ etc...** **-----END RSA PRIVATE KEY-----** **-----BEGIN CERTIFICATE-----** **MIICOzCCAaQCAQIwDQYJKoZIhvcNAQEEBQAwazELMAkGA1UEBhMCOBg NVetc...** **-----END CERTIFICATE-----**

9.4 Service Commands

9.4.1 AT+pVER – Software Version

Reports the RTX337x current software version.

Syntax:	**AT+pVER?**
Scope:	User level
Result code:	RTX337x-1_0 (example)
Example:	**AT+pVER?<cr>** RTX337x-1_0

9.4.2 AT+pLFLG – Logging Flags

Sets the flag that controls which type of information is logged.

Syntax:	**AT+pLFLG=str**
Parameters:	str = Decimal value of bit-enabled log type Bit 0 enables Typeless Messages Bit 1 enables Errors Bit 2 enables Warnings Bit 3 enables Informational Bit 4 enables Debug Messages Bit 5 enables Unknown Messages Bit 6 enables LOG-output to debug channel Bit 7 enables Technical Function Interface as debug channel. (LOG-output, if enabled, and additional debug information)
Default value:	127 - All logging enabled, but not written to Serial interface
Scope:	Operator level
Result code:	**OK** if legal value. Otherwise: **Error – LFLG value out of range**.
Example:	**AT+pLFLG=127<cr>** OK
Note:	The output enabled by bit 6 is never written to the log file and bit 7 enables writing data controlled by bit 0-6. For all possible debug information to the serial interface use AT+pLFLG=255.

Report the current value of the logging flag.

Syntax:	**AT+pLFLG?**
Scope:	User level
Result code:	127 (example)
Example:	**AT+pLFLG?<cr>** 127

9.4.3 AT+pFS – Reset to Factory Settings

Resets all configuration values to default values.

Syntax:	**AT+pFS=**
Parameters:	None.
Default value:	None.
Scope:	Operator level
Result code:	**OK**.
Example:	**AT+pFS=<cr>** OK
Note:	AT+pFIRMLIST, AT+pGWID and AT+pFIRMCRC is NOT reset.

Report all configuration values the user has access to.

Syntax:	**AT+pFS?**
Scope:	User level; TTY only, not usable from server.
Result code:	(current configuration)
Example:	**AT+pFS?<cr>** AT+pCODE? Permission code: 0X0000000000000000 Customer: No code available AT+pTIME? 2008/10/09 09.04.17 AT+pISP? 0W95551212 AT+pISP_BACKUP? AT+pWSPATH? rtx337x/nsservice/rtx337x.asmx AT+pWS? 1.2.3.4 AT+pUSR? | | | | | | | |
Note:	From server: AT+pSCRIPT?, AT+pINSIMAGE?, AT+pINSVOICE?, AT+pSCRIPTSTOR?, AT+pINSFONT? and AT+pINSSCRIPT? fulfills the same purpose.

9.4.4 AT+pCLRMESS – Clear the message buffer

Remove all messages stored for upload to the server.

Syntax:	**AT+pCLRMESS**
Parameters:	None.
Default value:	None.
Scope:	Operator level
Result code:	**OK**.
Example:	**AT+pCLRMESS<cr>** OK

9.4.5 AT+pTEST – Send Test Message

Send a test message to the server. The test message is sent via a TEST telegram to the server. This command will queue the telegram to be sent to the server, but communication error with the server will not be reported here. In case of error when connecting to the server, the error is logged.

Syntax:	**AT+pTEST=str** Length of str < 200 characters
Parameters:	Str=Testmessage. This text is used as the <Value> in the TEST telegram. "{" and "}" characters cannot be used.
Default value:	None.
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise **ERROR SAVING DATA IN DATABASE**.
Example:	**AT+pTEST=This is a test message<cr>** OK

9.4.6 AT+pLOGSIZE – Configure size of Internet log

Sets the size of the Internet portion of the log output

Syntax:	**AT+pLOGSIZE=str**
Parameters:	str = Integer in the range 7500 <= str <= 50000
Default value:	7500
Scope:	Operator level
Result code:	**OK** if legal value. Otherwise: **Error – LOGSIZE value out of range** or **Error - LOGSIZE value contains non numerical characters**.
Example:	**AT+pLOGSIZE=7500<cr>** OK
Note:	Changing the log size will clear the current Internet log

Report all configuration values the user has access to.

Syntax:	**AT+pLOGSIZE?**
Scope:	User level
Result code:	The currently configured log size.
Example:	**AT+pLOGSIZE?<cr>** 45000

9.4.7 AT+pLOG – Send Log to console

Send the log to the console output.

Syntax:	**AT+pLOG?**
Parameters:	None
Default value:	None
Scope:	Operator level
Result code:	(the log)
Example:	**AT+pLOG?<cr>** **** CONFIGURATION LOG **** 2004/11/23 15.28.06 LOG_INFO: operlevel - gwTiming changed to: I-0 2004/11/23 15.28.06 LOG_INFO: operlevel - userName changed to: 120102612287 etc… **** INTERNET LOG **** 2004/11/23 15.30.57 LOG_INFO : CInternetConnect - open Internet connection. 2004/11/23 15.30.57 LOG_INFO : CInternetCom - sending ordinary message to Internet server. etc…
Note:	Each log entry is limited to 511 characters. Truncated entries are marked, at the end, like this: **… Truncated!**

9.4.8 AT+pRFU – Remote Firmware Update

Upgrade firmware.

Syntax:	**AT+pRFU=str**
Parameters:	Str identifies the records with the firmware in the binary buffer, this data must be uploaded before the command is executed. The format is: Checksum(hex),name1,name2,…..,nameN Where names must be a maximum of 10 chars, and refer to binary data already uploaded and buffered.
Default value:	None.
Scope:	Operator level
Result code:	Status message showing whether the firmware and firmware command is accepted, if its not accepted details are logged.
Example:	**AT+pRFU=246d00fb,firm_aa,firm_ab,firm_ac,firm_ad<cr>** OK - attempting to perform firmware update **AT+pRFU=246d00fb,firm_aa,firm_ab,firm_ac,firm_ad<cr>** Error - Firmware not accepted
Note:	It is important that the database is not locked when the command is executed. For the format of the binary data see section 10.3, Uploading JavaScrips, Sound files and Images.

9.4.9 AT+pREQFU – REQuest a Firmware Update from the server

Ask the server to update the firmware.

Syntax:	**AT+pREQFU**
Parameters:	None
Default value:	None.
Scope:	Operator level
Result code:	OK
Example:	**AT+pREQFU** OK
Note:	Forces a connection to the server with a SENDRFU telegram. Whether anything more happens is up to the server.

9.4.10 AT+pFIRMLIST – Update the list of approved firmwares

Replace the list of approved hardware/software combinations currently stored with a new one.

Syntax:	**AT+pFIRMLIST=str**
Parameters:	Str identifies the record with the firmware list in the accompanying binary data. Str must be max 10 characters and consist of letters and digits only. For the format of the binary data see section 10.3, Uploading JavaScrips, Sound files and Images.
Default value:	None.
Scope:	Operator level
Result codes:	**OK** if success and **ERROR** otherwise.
Example:	**AT+pFIRMLIST=listname** OK
Note:	Stores the binary record identified by str in the database.

9.4.11 AT+pFIRMVALID – Verify that a given firmware is valid

Verify that a given firmware is valid for the current hardware/software combination

Syntax:	**AT+pFIRMVALID=str**
Parameters:	Str is an ASCII representation in hexadecimal of a firmware checksum.
Default value:	None.
Scope:	Operator level
Result code:	**Firmware is valid** - The checksum is valid. **ERROR - Firmware is invalid** – The firmware is not acceptable for this hardware. **ERROR - opening Firmware List** – The list could not be opened. **ERROR - Could not perform check –** Another error occurred.
Example:	**AT+pFIRMVALID=246d0fe4** Firmware is valid **AT+pFIRMVALID=246d0fee** ERROR – Firmware is invalid

9.4.12 AT+pFIRMCRC – CRC and length of the firmware

Sets the value of the CRC and length of the firmware, this is the values used during the Firmware verification at boot time

Syntax:	**AT+pFIRMCRC =str**
Parameters:	str=<crc>,<length> crc: This is the CRC-checksum of the firmware. length: This is the length in bytes of the binary firmware file.
Default value:	None.
Scope:	Operator level
Result code:	**OK** if legal value. Otherwise: **ERROR SAVING DATA IN DATABASE**

Get the current value of the CRC and length of the firmware.

Syntax:	**AT+pFIRMCRC?**
Parameters:	None
Default value:	None.
Scope:	Operator level
Result code:	The value of the CRC and the length of the image.
Example:	**AT+pFIRMCRC?** 2b5956e2,1962832

9.4.13 AT+pSCRIPT– Get the current RTX337x configuration

Get the current configuration in a script form, ready to be used to configure another device.

Syntax:	**AT+pSCRIPT?**
Parameters:	None
Default value:	None.
Scope:	Operator level
Result code:	The current configuration on script form.
Example:	**AT+pSCRIPT?** AT+pCODE=4598734752985 AT+pLCK AT+pISP=16110 AT+pISP_BACKUP= AT+pWSPATH= rtx337x/nsservice/rtx337x.asmx AT+pWS=80.63.27.150 AT+pUSR= AT+pPSW= AT+pDNS= AT+pPRXY= AT+pSAUT=0 AT+pCAUT=0 AT+pCACRT= AT+pCCRT= AT+pGW=I-0 AT+pGWR= AT+pINSDV=WEIGHT:ADBTWSType:FFFFFFFFFFFF-39121440-weight- - - -70-50-65-15:I-0:I-0

	AT+pOPER=operlevel:operpass AT+pUSER=userlevel:userpass AT+pLFLG=127 AT+pFALLBACK=0 AT+pFACTORYSCRIPT= AT+pPATIENTSCRIPT= AT+pNORMALSCRIPT= AT+pTECHSCRIPT= AT+pMODE=FACTORY AT+pRESETCOUNT=0 AT+pVOLUME=0,2,4,6,8,10,12,14,16,18,6:VolAdj AT+pFIRMCRC=610e660c,2702520 AT+pMBEG1= AT+pMBEG1_VC= AT+pMBEG2= AT+pMBEG2_VC= AT+pMPRED1= AT+pMPRED1_VC= AT+pMPRED2= AT+pMPRED2_VC= AT+pMPRED3= AT+pMPRED3_VC= AT+pMEND= AT+pMEND_VC= AT+pMCDBEG= AT+pMCDBEG_VC= AT+pMCDEND= AT+pMCDEND_VC= AT+pISP_VC= AT+pBACKLIGHT=90,60,1000,2 AT+pAUTODIAL=1 AT+pFONTSIZE= AT+pBACKGROUND=back1:back2 AT+pBTLOCALNAME=RTX337x AT+pLOGSIZE=7500 AT+pGPRSAPN= AT+pGPRSIP=0.0.0.0 AT+pPINCODE= AT+pGPRSHEADCOMP=0 AT+pGPRSDATACOMP=0 AT+pUNLCK
Note:	Items with binary content, phrase files, image files, font files, script store files and JavaScripts are *not* included in the list.

9.4.14 AT+pACTIVATE – Activate Script

Activate a given JavaScript.

Syntax:	**AT+pACTIVATE=str**
Parameters:	Str has the format: ScriptId,priority,activatorId,nrParam,param0,param1,… Where ScriptId is the name of the JavaScript to activate (a string – max 10 letters or digits), priority is the priority of the script (an integer – 0 lowest priority), activatorId is identification of the activator (a string – max 50 letters or digits), nrParam is the number of parameters for the JavaScript (an integer smaller than 26 and param0, param1, … are the parameters (double values)).
Default value:	None.
Scope:	User level
Result code:	**OK** if legal parameter. Otherwise **Syntax Error . "description".**
Example:	**AT+pACTIVATE=Default,0,sys,0** OK

9.4.15 AT+pRESETCOUNT – Set or request the reset count

Set the reset counter.

Syntax:	**AT+pRESETCOUNT=number**
Parameters:	The number to set the counter to The value must be in the range 0 - 2^31
Default value:	0
Scope:	Operator level
Result code:	**OK** if legal parameter. Otherwise **ERROR - Argument must be numeric, ERROR SAVING DATA IN DATABASE** or **ERROR - Argument out of legal range.**
Example:	**AT+pRESETCOUNT=0** OK

Get the current value of the reset counter

Syntax:	**AT+pRESETCOUNT?**
Parameters:	None.
Default value:	0
Scope:	User level
Result code:	The current value of the counter
Example:	**AT+pRESETCOUNT?** 0

9.4.16 AT+pTIMEBACKUP – Get the back up time

Get the current value of backup time in the database, this is the time that will be restored if the device is left without power until the battery driven clock fails.

Syntax:	**AT+pTIMEBACKUP?**
Parameters:	None
Default value:	None.
Scope:	User level
Result code:	The value of the current backup time, as a UNIX timestamp.
Example:	**AT+pTIMEBACKUP?** 1141733143

9.4.17 AT+pSTATUS – Get the status word

Syntax:	**AT+pSTATUS?**
Parameters:	None
Default value:	0 (no errors)
Scope:	User level
Result code:	The value of the current status, decimal representation of binary flags: Bit 0: Firmware CRC check failed Bit 1: Memory check failed. Bit 2: No longer used. Bit 3: IO error while accessing file system. Bit 4: Inconsistency detected in storage of phrase files. Bit 5: BT module failed. Bit 6: BT address in flash corrupted. Bit 7: BT address mismatch. Bit 8: Modem failed. Bit 9: RTC failure.
Example:	**AT+pSTATUS?** **8**
Note:	The status flag is cleared whenever the flag has been successfully uploaded to the server. A BT module failure will always lead to a BT address mismatch.

9.4.18 AT+pHOSTSTATUS – Get the host connection status

Get the status for the last Host Server connection attempt.

Syntax:	**AT+pHOSTSTATUS?**
Parameters:	None
Default value:	0 (No error)
Scope:	User level
Result code:	The value of the current Host Server status as a decimal value of the binary value, the format of the value is: Field 1: Bit 0-3: Modem connection status value: • 0 – Success • 1 – Error • 2 – Busy • 3 – No answer (note modem can only report this if phone number dialed ends with '@') • 4 – Ringing … Ringing (other end does not pick up phone after two rings, note this is only active if ATX5 is sent to the modem and it will cause this to be reported instead of No Answer). • 5 – No carrier • 6 – No dial tone • 7 – Line in use • 8 – No line • 9 – SIM PUK-code required. Field 2: Bit 4: ISP Connection • 0 – Success • 1 – Error (note: This will not be set if pre-dial fails otherwise if the modem connection status is not 0, this will be set as well). Field 3: Bit 5: Host Server communication error • 0 – Success • 1 – Error (note: This will only be set if the first 2 fields are 0) Field 4: Bit 6-8: GSM/GPRS modem initialization status value: • 0 – Success • 1 – No SIM card present • 2 – Wrong PIN code for SIM card • 3 – No PIN code available for SIM card • 4 – SIM card requires PUK code
Example:	**AT+pHOSTSTATUS?** 24
Note:	Before every connection attempt this value is cleared.

9.4.19 AT+pLASTMEAS – Get the last measurement

Get the last received measurement. If no measurements have been received this is empty.

Syntax:	**AT+pLASTMEAS?**
Parameters:	None
Default value:	None.
Scope:	User level
Result code:	The value of the last measurement received from an external device, this has the format: <DeviceId>-<Value>. Where <DeviceId> and <Value> refers to the content of these tags in the XML telegram, for details about this format please see section 10.3, Uploading JavaScrips, Sound files and Images.
Example:	**AT+pLASTMEAS?** 00A0960D4F08:41-<SubDev>Ws1</SubDev><SubValue>+001.00 kg</SubValue>

9.4.20 AT+pLASTRESP – Get the userresponse to the last measurement.

Get the user response, for the last received measurement value, if no user response was sent this it empty.

Syntax:	**AT+pLASTRESP?**
Parameters:	None
Default value:	None.
Scope:	User level
Result code:	The value of the user response to the last measurement (see AT+pLASTMEAS command), this has the format: <DeviceId>-<Value>. Where <DeviceId> and <Value> refers to the content of these tags in the XML telegram, for details about this format please see chapter 12, XML.
Example:	**AT+pLASTRESP?** 00A0960D4F08:41-mister Miller W: 1 Time: 1149088094 Type: 1 U: 70 L: 50 Did you wake short of breath during the night? No
Note:	If no user response message was sent to the last received measurement this will be empty.

9.4.21 AT+pTTYENABLE – Get the TTY mode

Syntax:	**AT+pTTYENABLE?**
Parameters:	None
Default value:	None.
Scope:	User level
Result code:	Returns the current value of the TTY status, the possible values are: • Ymodem upload = -2 • Device Interface = -1 • Technical interface ≥ 0 (a positive number is a timeout in seconds, 0 is no timeout)
Example:	**AT+pTTYENABLE?** -1

9.4.22 AT+pPRODINFO –Production Information

Report production information.

Syntax:	**AT+pPRODINFO?**
Scope:	User level
Result code:	Production information and test date.
Example:	**AT+pPRODINFO?<cr>** ORDER CODE : 50100004 DMR : 05 S/N : RTX33700000034 TESTDATE : 2007/10/23 10:21:03

9.4.23 AT+pMODEL – Get The Hardware Model

Report the hardware model.

Syntax:	**AT+pMODEL?**
Scope:	User level
Result code:	Returns the hardware model. The possible values are: • RTX3370 • RTX3371
Example:	**AT+pMODEL?** RTX3370

9.5 Testing Commands

9.5.1 AT+pSTOP_SYSTEM – Enter Test Mode

Stops normal operation and enters test mode.

Syntax:	**AT+pSTOP_SYSTEM**
Parameters:	None.
Default value:	None.
Scope:	Operator level; TTY only, not usable from server.
Result code:	**OK**.
Example:	**AT+pSTOP_SYSTEM<cr>** OK
Note:	Care should be taken to ensure the system is started again, if a logout is performed (includes inactivity timeout) the system will not start again automatically.

9.5.2 AT+pSTART_SYSTEM – Restart from Test Mode

Exits test mode and restarts normal operation.

Syntax:	AT+pSTART_SYSTEM
Parameters:	None.
Default value:	None.
Scope:	Operator level; Test mode
Result code:	OK.
Example:	AT+pSTART_SYSTEM<cr> OK

9.5.3 AT+pTEST_BUTTON – Button Test

Tests user-accessible buttons. This command can only be used when the system has been stopped (STOP_SYSTEM).

Syntax:	**AT+pTEST_BUTTON=str**
Parameters:	str = Button:Timeout where Button: the button to be tested 0: Left 1: Right 2: Select Up 3: Select Down 4: Volume Up 5: Volume Down 6: Info button Timeout: time in seconds, max 120 seconds
Default value:	None
Scope:	Operator level; Test mode
Result code:	**OK** if the button is pressed within the timeout period. **Timeout** if the button was not activated during the specified time. **Syntax Error** if the command was not typed correctly.
Example:	**AT+pTEST_BUTTON=1:5<cr>** OK

9.5.4 AT+pMODEM – Modem Pass-Through

Allows pass-though of commands to the modem. This command can only be used when the system has been stopped (STOP_SYSTEM).
After the execution of the command, a direct link to the modem is established. To exit this mode, type '|'.

Syntax:	**AT+pMODEM**
Parameters:	None
Default value:	None
Scope:	Operator level; Test mode
Result code:	None
Example:	**AT+pMODEM<cr>** **ATI1** 255 **|** *These commands will query the modem's "I1" register and then return to test mode.*

9.5.5 AT+pSTOP_WATCHDOG – Watchdog Test

Stops the watchdog to test response. System should restart in response. This command can only be used when the system has been stopped (STOP_SYSTEM).

Syntax:	**AT+pSTOP_WATCHDOG**
Parameters:	None
Default value:	None
Scope:	Operator level; Test mode
Result code:	**OK**
Example:	**AT+pSTOP_WATCHDOG<cr>** OK

9.5.6 AT+pBACKLIGHTTST – Backlight Test

Tests the LCD backlight. This command can only be used when the system has been stopped (STOP_SYSTEM).

Syntax:	**AT+pBACKLIGHTTST=str**
Parameters:	str = x to control the LCD backlight as follows. 0 Off 1 Low intensity 2 High intensity
Default value:	None
Scope:	Operator level; Test mode
Result code:	**OK** or **Syntax Error** if the command was not typed correctly.
Example:	**AT+pBACKLIGHTTST=1<cr>** OK

9.5.7 AT+pDISPLAY – LCD Test

Displays test patterns on the LCD. This command can only be used when the system has been stopped (STOP_SYSTEM).

Syntax:	**AT+pDISPLAY=str**
Parameters:	str = x, where x defines the pattern to be displayed. 0 Display test off. 1 All pixels black 2 All pixels white 3 All pixels red 4 All pixels green 5 All pixels blue
Default value:	None
Scope:	Operator level; Test mode
Result code:	**OK** or **Syntax Error** if the command was not typed correctly.
Example:	**AT+pDISPLAY=1<cr>** OK

9.5.8 AT+pIRDATEST – Test IrDa communication

Puts the IrDa communication channel into an echo mode, all received characters are incremented by one and returned immediately. This command can only be used when the system has been stopped (STOP_SYSTEM). To exit this mode either type ‘{’ or wait 10 seconds without IRDA activity. Setup for IRDA is 9600 baud, 1 start bit, 1 stop bit and none parity.

Syntax:	**AT+pIRDATEST**
Parameters:	None
Default value:	None
Scope:	Operator level; Test mode
Result code:	**OK**
Example:	**AT+pIRDATEST<cr>** OK

10 JavaScripts and MMI

RTX337x has support for JavaScript conforming to the ECMAScript-262 specification. In addition to the core JavaScript language, context for the RTX337x is added in the form of a RTX337x customized JavaScript object.

The customized RTX337x JavaScript object name is gw.

JavaScript allows for flexible control of certain aspects of the system, in particular with regards to user interaction via the RTX337x buttons, the RTX337x display and the RTX337x speaker.
The display on the RTX337x can be used to show questions that can be answered by pushing the buttons below the display. A speaker can verbally support the information shown on the display. After the RTX337x has received the measured values and the answers, it generates a call and conveys the data to a remote facility via a public telephone line. In some cases, the RTX337x is programmed to deliver data at a specific time of day. The delivery of data can automatically be repeated if the attempt to deliver data fails.

JavaScripts are written by the user and are entered into the system by means of AT+p commands. Activation of a given JavaScript is connected to a given RTX337x event by means of AT+p commands. Configurable texts and images to be displayed, and configurable phrases to be voiced are entered into the system by means of AT+p commands.

References to JavaScripts are by means of names.
References in JavaScripts to configurable texts to be displayed, and configurable phrases to be voiced, are by means of names.
Names are ASCII strings consisting of letters and/or digits. Maximum number of characters is 10.
The same name can be used for a text to be displayed, and a phrase to be voiced.

NOTE: Anonymous functions in JavaScript are not supported.

10.1 Making sound files

The RTX337x uses a compressed sound format called SBC (Sub-Band Codec). The RTX337x has some requirements to the input format of the Wave files before they are compressed to SBC:

- Sample rate must be 16000 Hz
- File must be in Mono
- 16 bit PCM format

The following settings for the sbc_encoder program works well:
"`sbc_encoder -l16 -n8 -r60000`" This seems to give good results compared to file size but there might be a better quality/space relationship. This is also influenced by the available memory space on the RTX337x, how many minutes of sound the application requires and, if the files are meant to be uploaded through the Host Server connection, time/traffic cost.

If combined phrases are to be used, i.e. using multiple files to form a complete sentence, care must be taken to ensure the sound files are ending with silence and that the silence has the appropriate duration to make the completed sentence sound natural.

The file format used by the RTX337x includes support for putting the corresponding text phrase in the same file as the SBC encoded audio. This means that some additional processing of the SBC files is required before uploading them to the RTX337x. The format of the combined files on the RTX337x is:

a) A SBC file with the sound track (speech).
b) A txt string with the Latin-1 text (only for built-in font) or UTF-8 text (only for TrueType fonts) for the above SBC file

The format of the content must be:

- A header consisting of two fields of 4 Bytes (Little Endian format) each containing the length in bytes of each of the sub records followed by the two records in the order a, b.

10.2 Making image files

The RTX337x uses the compressed image format JPEG.
The RTX337x has a limitation to the format of the image files so they have to be without thumbnail information.

NOTE: JPEG images created with the standard "Imaging for Windows" bitmap editor is not accepted by the RTX337x.

10.3 Uploading JavaScripts, Sound files, Images and fonts

The types of files used for the MMI on the RTX337x are:

- Text files containing JavaScript code
- Phrase files in the RTX337x format, see section 10.1, Making sound files.
- JPEG files containing image.
- TrueType Font files

Before a file can be uploaded to the RTX337x, either through the Technical function interface or through the Host Server, it has to be packed into a format for upload. This format can include one or more records in the file. Each record has the format:

1. The first field is the length of the content field in bytes. Size – 4bytes (Little Endian format)
2. The next field is the name (in ASCII letters and/or digits) as a string terminated with binary zero. This field identifies the record and must be unique. Size – 11 bytes, right padded with binary zero(s).
3. The last field of the record is the content. Size – length bytes

The last record has a length field, which is zero and no name field.

The content of the content field is the binary data, There are no restrictions on the content.

After this file has been uploaded (either through AT+pBINARYUPLOAD or through the SOAP interface) the content can be referenced either by the AT+pINSSCRIPT, AT+pINSVOICE, AT+pINSIMAGE or by the AT+pINSFONT command to permanently store them in the RTX337x.

> NOTE: The uploaded data is temporary until stored through a command, it is only stored until the next batch completes i.e. until either the first AT command is executed or if AT+pLCK is used as the first command when the following AT+pUNLCK is executed.
> There is no safeguard regarding the format of the data saved using AT+pINSSCRIPT, AT+pINSVOICE, AT+pINSIMAGE or AT+pINSFONT, so care should be taken to issue these commands with the correct names.
> It is not recommended to upload binary files larger than 700-800 KByte and even smaller files can give trouble if the RTX337x has been configured several times with large files without reboot, since the temporary memory area get fragmented during use.

10.4 Using texts, images and buttons

Scripts are used to control the look and feel of the user MMI. This section describes the requirements and constraints of the MMI.

10.4.1 Text and image display positions

A number of predefined windows, known as *text and image positions*, exist. Each text and image position has a fixed width and height beyond which text cannot be written. The text and image positions are used to show general information. They may contain one or more lines of text using fonts. Flashing text is also supported. Flashing and font selection is supported by an entire text, i.e. it is not possible to have individual words in text flashing or using different fonts. Text can be centered, right or left aligned. Margins can be defined to control the alignment. An image will fill the selected window independent of the original size and aspect ratio of the image. Writing to a text and image position automatically deletes any text or image already present at that position.

The table below shows the number of lines each text and image position is capable of showing. The number of characters allowed on each line varies with the text, as the font is a proportional font. This table is only true for the build in font, for installed TrueType fonts this will change.

Text and image position	Lines of text				Image size in pixels
	Normal font	Big font	Small font	Small slim font	
Full window frame	7	5	8	8	280 x 192
Half part window frame	3	2	4	4	280 x 96
Two-thirds part window frame	3	4	5	5	280 x 128
Third part window frame	2	1	2	2	280 x 64
Menu Header	1	1	1	1	280 x 32
Large Menu Header	2	1	2	2	280 x 64
X-Large Menu Header	3	2	4	4	280 x 96

The following sections describe the various text positions.

10.4.1.1 One frame window

The full window frame fills put the entire display area dedicated for text and images. Text will be displayed as one string, which may wrap to use multiple lines. Images will fill out the entire display area dedicated for text and images. If there is no more room for text in a chosen frame the text will be cut off.

Have you had any events of hypoglycemia within the last seven days?		**Full window frame** 1 string of text and/or an image (280 x 192 pixels).
YES	NO	Soft button texts
The full window frame layout can show one string, which may wrap to use multiple lines of text and/or one image. Flashing text is allowed. The font may be big, small, small slim or normal. Text can be centered, right or left aligned.		

10.4.1.2 Two frame windows

The half part window frame is either the upper or lower half part of the display area dedicated for text and images. Text or images are displayed only in the defined area. Text may be wrapped to use multiple lines. Frame limits are not visible in the display. If there is no more room for text in a chosen frame the text will be cut off.

You have had 4 events of hypoglycemias within the last 7 days		**Upper half part window frame** 1 string of text and/or an image (280 x 96 pixels).
Your weight is 214 pounds.		**Lower half part window frame** 1 string of text and/or an image (280 x 96 pixels).
OK		Soft button texts
The half part window frames (upper and lower) can show one string of text and/or one image. The font may be big, small, small slim or normal. Word wrapping and flashing text is allowed. Text can be centered, right or left aligned.		

The two-thirds part window frame is either the upper or the lower two-thirds of the display area dedicated for text and images. Text or images are displayed only in the defined areas. Text may be wrapped to use multiple lines. If there is no more room for text in a chosen frame the text will be cut off.

You have had 4 events of hypoglycemias within the last 7 days		**Upper two-thirds part window frame** 1 string of text and/or an image (280 x 128 pixels).
Tuesday 30 October 2007		Lower third part window frame (see 10.4.1.3) 1 string of text and/or an image (280 x 64 pixels).
		Soft button texts
The two-thirds part window frames (upper or lower) can show one string of text and/or one image. The font may be big, small, small slim or normal. Word wrapping and flashing text is allowed. Text can be centered, right or left aligned. The two-thirds window can be combined with the third part window frame.		

Lower two-thirds part window frame:

Please take your blood pressure		Upper third part window frame (see 10.4.1.3) 1 string of text and/or an image (280 x 64 pixels).
		Lower two-thirds part window frame 1 string of text or an image (280 x 128 pixels).
OK		Soft button texts
The two-thirds part window frames (upper or lower) can show one string of text and/or one image. The font may be big, small, small slim or normal. Word wrapping and flashing text is allowed. Text can be centered, right or left aligned. The two-thirds window can be combined with the third part window frame.		

10.4.1.3 Three frame windows

The third part window frame is either the upper, middle or lower third part of the display area dedicated for text and images. Text or images are displayed only in the defined area. If there is no more room for text in a chosen frame the text will be cut off.

		Upper third part window frame 1 text string and/or an image (280 x 64 pixels)
Did you take your medication today?		**Middle third part window frame** 1 text string and/or an image (280 x 64 pixels)
Monday 30 October 2007.		**Lower third part window frame** 1 text string and/or an image (280 x 64 pixels)
YES	NO	Soft button texts
The third part window frames (upper, middle or lower) can show one string of text and/or one image. The font may be big, small, small slim or normal. Word wrapping and flashing text is allowed. Text can be centered, right or left aligned. Upper and lower third part window frames can be combined with the lower or upper two-thirds window frames.		

10.4.1.4

10.4.1.5 Menu Headers

The menu header shows one string only, and is intended for use as a header located above a menu. There is only room for one line of text in the menu header.

Technical Menu		**Menu header** 1 string of text and/or image for the menu header.
		This space is used for the menu itself, see 10.4.3
Enter		Soft button texts
The menu header can show one string only, and is intended for use as a header located above the MENU layout. Word wrapping is not allowed. The font may be big, small, small slim or normal and flashing is allowed. Text can be centered, right or left aligned.		

The large menu header shows one string only, and is intended for use as a header located above a menu. Text may be wrapped to use multiple lines. Number of lines of text in the large menu header is dependent on the font, see table in section 10.4.1. If there is no more room for text the text will be cut off.

<table>
<tr><td colspan="2">How would you describe your health in the last couple of days?</td><td><b>Large Menu header</b>
1 string of text and/or image for the menu header.</td></tr>
<tr><td colspan="2"></td><td>This space is used for the menu itself, see 10.4.3.</td></tr>
<tr><td>Select</td><td></td><td>Soft button texts</td></tr>
<tr><td colspan="3">The large menu header can show one string only, and is intended for use as a header located above the MENU layout. Word wrapping is allowed. The font may be big, small, small slim or normal and flashing is allowed. Text can be centered, right or left aligned.</td></tr>
</table>

The X-large menu header shows one string only, and is intended for use as a header located above a menu. Text may be wrapped to use multiple lines. Number of lines of text in the X-large menu header is dependent on the font, see table in section 10.4.1. If there is no more room for text the text will be cut off.

<table>
<tr><td colspan="2">How do you feel your blood glucose level has been within the last seven days?</td><td><b>X-Large Menu header</b>
1 string of text and/or image for the menu header.</td></tr>
<tr><td colspan="2"></td><td>This space is used for the menu itself, see 10.4.3.</td></tr>
<tr><td>Select</td><td></td><td>Soft button texts</td></tr>
<tr><td colspan="3">The X-large menu header can show one string only, and is intended for use as a header located above the MENU layout. Word wrapping is allowed. The font may be big, small, small slim or normal and flashing is allowed. Text can be centered, right or left aligned.</td></tr>
</table>

10.4.2 Word wrapping

When text is written to any of the text positions it automatically wraps to the next line if the text is too long to fit in one line. Wrapping always occurs in the space between words. If the height of the text position cannot contain the resulting number of lines the excess text is clipped. Similarly, if the line cannot be wrapped, e.g. if it contains no spaces, the end of the line is clipped. Clipping is only to be done at the edge of the window, right and bottom margins will be overwritten if text can’t fit.

When wrapping to the next line occurs all leading spaces are removed from the remaining text. This may be used to control the point of wrapping.

E.g.: the text “XXX YY ZZZZZ” must wrap before ZZZZZ but as YY has a variable length the wrapping point will also vary. To fix this the text may be changed to
“XXX YY   ZZZZZ”. This ensures that ZZZZZ will always wrap to the next line and centered correctly as any leading space before ZZZZZ is removed.

10.4.3 Menus

The menu layout is used to display menu structures. It consists of a window for the menu header (see section 10.4.1.5), a window for the menu items and two windows for the soft button texts.

<table>
<tr><td colspan="2">Technical Menu</td><td>Menu header</td></tr>
<tr><td colspan="2">Selection 1</td><td rowspan="5"><b>Menu or Info lines</b>
3, 4 or 5 lines of menu items visible at a time (Depending on the size of the menu header).</td></tr>
<tr><td colspan="2">Selection 2</td></tr>
<tr><td colspan="2">Selection 3</td></tr>
<tr><td colspan="2">Selection 4</td></tr>
<tr><td colspan="2">Selection 5</td></tr>
<tr><td>Enter</td><td></td><td>Soft button texts</td></tr>
<tr><td colspan="3">Font selection is not supported in the menu items. Menu entries may only contain one line of text.
Two different functional configurations exist (Menu and Info):
When configured for “Menu”, pressing the scroll buttons automatically selects the next or previous menu item. When scrolling, hidden selections (only 3, 4 or 5 are visible at a time) enter the display until the selection bar is either in the top or bottom of the menu.
When configured for “Info” pressing the scroll buttons scrolls all lines with no graphic indication. When scrolling, hidden text will become visible line after line.
Text can be centered, right or left aligned.</td></tr>
</table>

10.4.4 Using buttons

The buttons on the front may be utilized by the JavaScript code.

10.4.4.1 Showing soft buttons text

The left and right buttons have corresponding field on the display for soft button texts. These fields may be written to using the **ShowButtons()** command.

10.4.4.2 Activating buttons

The buttons must be activated before they will register the user’s presses. Each button can be activated individually using **ActivateButtons()**.

10.4.4.3 Deactivating buttons

Pressing any activated button automatically deactivates all buttons. The buttons may also be deactivated using **ActivateButtons()**.

10.4.5 Configuration of text and image

RTX337x JavaScript object methods for displaying text and image have a configuration parameter (cnf) with the following possible values:

NOTE: bits are numbered right to left, so bit 0 is the rightmost bit (23…0).

Field	Bits	Value	Description
Position	0:3		This field identifies the display window. For a detailed description of the supported windows see 10.4.1. The position field is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage(), ShowTextAndImageL(), ShowImage() and ClrText() commands.
		0	Full window frame
		1	Upper half part window frame
		2	Lower half part window frame
		3	Upper third part window frame
		4	Middle third part window frame
		5	Lower third part window frame
		6	Menu header
		7	Large menu header
		8	X-Large menu header
		9	Upper two-thirds part window frame
		10	Lower two-thirds part window frame
font	4:7		This field controls the font used when writing to the display. The font is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage() and ShowTextAndImageL() commands.
		0	Normal font
		1	Big font
		2	Small font
		3	Small slim font
Display	12:15		This field controls flashing of the display text. If enabled when writing a string to the display the entire string will blink. This parameter is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage(), ShowTextAndImageL(), ShowMenu() and ShowMenuL() commands.
		0	Normal
		1	Flashing
Menu	16:19		This field controls menu behavior. This parameter is disregarded by all but the ShowMenu() and ShowMenuL() commands.
		0	Menu 5 lines (Used with Menu header)
		1	Menu 4 lines (Used with Large menu header)
		2	Info 5 lines (Used with Menu header)
		3	Info 4 lines (Used with Large menu header)
		4	Menu 3 lines (Used with X-Large menu header)
		5	Info 3 lines (Used with X-Large menu header)
Alignment	20:23		This field controls the horizontal text alignment in display

			windows. This parameter is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage(), ShowTextAndImageL(), ShowMenu() and ShowMenuL() commands.
		0	Text is centered.
		1	Text is aligned to the left.
		2	Text is aligned to the right.
Language	24:27		This field controls languages that need special treatment. This parameter is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage(), ShowTextAndImageL(), ShowMenu(), ShowMenuL(), ShowButtons() and ShowButtonsL() commands.
		0	Default
		1	Arabic shaping (Forced right to left direction)
Direction	28		This field controls the writing direction. This parameter is disregarded by all but the ShowText(), ShowTextL(), ShowTextAndImage(), ShowTextAndImageL(), ShowMenu(), ShowMenuL(), ShowButtons() and ShowButtonsL() commands.
		0	Default (Left to right)
		1	Right to left

For the built-in font 4 fonts sizes are available:

Normal font (bold):
28 pixels of 0.219 mm = 6.132 mm => 6.132 / 25.4 = 0.2414 inch => 0.2414 * 72 = 17.38pt

Big font (bold):
36 pixels of 0.219 mm = 7.884 mm => 7.884 / 25.4 = 0.3104 inch => 0.3104 * 72 = 22.35pt

Small font (bold):
24 pixels of 0.219 mm = 5.256 mm => 5.256 / 25.4 = 0.2069 inch => 0.2069 * 72 = 14.90pt

Small slim font:
24 pixels of 0.219 mm = 5.256 mm => 5.256 / 25.4 = 0.2069 inch => 0.2069 * 72 = 14.90pt

Be aware that font size is not the height of a single capital letter. It also leaves space for underlined letters and diacritical marks.

10.5 TrueType fonts

It is possible to install TrueType fonts on the RTX337x.

Two font names are reserved as default fonts:

- Bold.ttf
- Normal.ttf

When installed, Bold.ttf will replace the default Normal font, Big font and Small font.
When installed, Normal.ttf will replace the default Small slim font.

Any other file names, containing the extension .ttf, can be used, but they must be assigned with AT+pASSIGNFONT to specify which build in the font to replace.
The default sizes of the TrueType fonts can be changed with the AT+pFONTSIZE command. This setting has no effect on the built-in fonts.

See font usage in section 10.4.

10.5.1 Installing TrueType fonts

Use the AT+pBINARYUPLOAD (see section 9.1.17) command to load the fonts to the RTX337x memory. The ttf font files must be wrapped with size and name information as described in section 10.3. Use the following commands to install fonts:

- AT+pLCK
- AT+pINSFONT=font1.ttf
- AT+pINSFONT= font2.ttf
- AT+pINSFONT= font3.ttf
- AT+pINLCK

(See section 9.1.25)

Then assign fonts with AT+pFONTASSIGN.
This example assign fonts like this: Normal font = font3.ttf, Big font = font1.ttf, Small font = font2.ttf, Small slim font = font1l.ttf

- AT+pFONTASSIGN=font3.ttf, font1.ttf, font2.ttf, font1.ttf

(See section 9.1.27)

NOTE: When fonts are installed XML telegrams are generated with UTF-8 encoding.

LEGAL NOTICE: Be sure to have proper license to the used fonts.

10.5.2 Removing TrueType fonts

Installed TrueType fonts can be removed with:

- AT+pRMFONT=Bold.ttf
- AT+pRMFONT=Normal.ttf

(See section 9.1.26)

10.5.3 Changing TrueType font size

The default TrueType font sizes can be changed with the AT+pFONTSIZE command.

Example: AT+pFONTSIZE=23,31,20,19 sets:

- ➢ Normal font size to 23 (default 28)
- ➢ Big font size to 31 (default 36)
- ➢ Small font size to 20 (default 24)
- ➢ Small slim font size to 19 (default 24)

The current settings can be read with the AT+pFONTSIZE? command.

(See section 9.1.27.)

10.5.4 Checking installed fonts

Use AT+pINSFONT? to see installed fonts.
(See section 9.1.25)

10.5.5 Using TrueType fonts

When installed, the fonts are used in the same way as the built-in fonts they are replacing. RTX337x JavaScript object methods for displaying text and image have a configuration parameter (cnf) with values shown in section 10.4.5.

Special features are the 'Language' and 'Direction' settings. With 'Language' set to 1, the text is expected to be Arabic. Arabic text is parsed to a shaping routine and reversed to a right-to-left direction.
With Direction set to 1 (Language 0), only the character reversion is made (no shaping).

10.5.6 File save format

If any TrueType fonts are installed, the RTX337x interprets JavaScript files as UTF-8 formatted text. Therefore all scripts must be saved in this format. Otherwise characters in the range from 128 to 255 to be displayed will not be shown correctly.
If no TrueType fonts are installed, JavaScript files are interpreted as Latin-1 text and must be saved in this format.

NOTE: The UTF-8/Latin-1 format also holds for sound files where the file consists of compressed sound and text. The text part must have the correct format. See section 10.1 for sound file details.

10.6 Script memory

An area for storing data between JavaScript sessions or different scripts is provided. This data area is accessed through 10 character ASCII names.

The intent of this feature is to allow scripts to store the last time they were activated or the last time some specific event/script ran. The purpose is to support features like not asking questions more than once a day, have scripts verify whether other scripts have been run (measurements performed and so forth).

10.7 User Response Message from Script

From inside the script a temporary memory area is available for building a message that can be sent to the server.
This user response memory is cleared when the script is started.
It is possible to add information to this message several times during the script execution and then send the complete message when all information is collected.
The message type of this message is: USERRESPONSE. See table in section 12.2 for more information on message types.

10.8 Customized JavaScript Methods

The table below lists the customized JavaScript methods, subsequently the methods will be described.

ClrScr()	GetMBeg1()
GetText()	GetMBeg2()
ShowText()	GetMPreD1()
ShowTextL()	GetMPreD2()
ShowTextAndImage()	GetMPreD3()
ShowTextAndImageL()	GetMEnd()
ShowImage()	GetMCDBeg()
ClrText()	GetMCDEnd()
ShowMenu()	GetISP()
ShowMenuL()	SetMBeg1()
ClrMenu()	SetMBeg2()
ShowButtons()	SetMPreD1()
ShowButtonsL()	SetMPreD2()
ClrButtons()	SetMPreD3()
ActivateButtons()	SetMEnd()
PlaySound()	SetMCDBeg()
WaitForEvent()	SetMCDEnd()
ClrResp()	SetISP()
AddToResp()	SetISPBck()
SendRespToServer()	ResetModem()
GetLastMeasurement()	ATModemCommand()
GetLastResp()	GetConfigFromHost()
GetTime()	RegisterOnHost()
SetTime()	GetTestFromHost()
EnableConfiguration()	ForceConnect()
DisableConfiguration()	Connect()
Reset()	GetHostConnectionStatus()
GetScriptName()	GetStatus()
GetActivatorId()	GetCurrentPeripherals()
GetActivatorType()	ActivateScript()
GetNrScriptPars()	Log()
GetScriptPar()	SetFACTORYMode()
SetTimeout()	SetPATIENTMode()
ClrTimeout()	SetNORMALMode()
ClrAllTimeouts()	SetBacklight()
LEvent()	GetSaveData()
REvent()	SetSaveData()
SEvent()	InsertDevice()
IEvent()	UpdateDevice()
PHRASEEvent()	RemoveDevice()
HOSTEvent()	SetDeviceTiming()
DIALEvent()	SetDeviceSupervision()
CONNECTEvent()	GetDeviceType()
DISCONNECTEvent()	GetDeviceInfo()

PACKAGEEvent()	GetDeviceTiming()
TIMEOUTEvent()	GetDeviceSupervision()
EXITEvent()	GetSignalQuality()
StopSystem()	GetModel()
StartSystem()	GetVer()

10.8.1 ClrScr()

Clear the display

Synopsis:	ClrScr()
Arguments	None
Returns	Returns 0 (success)
Description	Clears the entire RTX337x display
Example	result = gw.ClrScr();

10.8.2 GetText()

Get text

Synopsis:	GetText(txtName)
Arguments	TxtName The name of the text.
Returns	Returns the text identified by txtName. Returns empty string in case of error.
Description	Get the text identified by txtName.

Example	txtName = "name";
	text = gw.GetText(txtName);

Note	The name of the text = Voicefilename (name of the file that contains text and/or sound)

10.8.3 ShowText()

Show text on the display

Synopsis:	ShowText(cnf, txtName)
	or
	ShowText(cnf, txtName, leftMargin, rightMargin, topMargin, bottomMargin)

Arguments	cnf How to show the text. txtName The name of the text. Optional (All or none) leftMargin Left margin in pixels. rightMargin Right margin in pixels. topMargin Top margin in pixels. bottomMargin Bottom margin in pixels.
Returns	Returns 0 for success and –1 for error.
Description	Shows the text identified by txtName on the display, existing text and/or image in the specified window is cleared. The text is shown on the display in accordance with the margins and the cnf parameter.

	See also section 10.4.5, Configuration of text and image. Text will automatically wrap to fit but carriage return can also be used to force line breaks.

Examples	cnf = 1;
	txtName = "name";
	result = gw.ShowText(cnf, txtName);
	
	or
	
	cnf = 1;
	txtName = "name";
	result = gw.ShowText(cnf, txtName, 20, 0, 20, 0);

Note	The name of the text = Voicefilename (name of the file that contains text and/or sound)

10.8.4 ShowTextL()

Show literal text on the display

Synopsis:	ShowTextL(cnf, text)
	or
	ShowTextL(cnf, text, leftMargin, rightMargin, topMargin, bottomMargin)	

Arguments	cnf	How to show the text.
	text	The text to show.
		Optional (All or none)
	leftMargin	Left margin in pixels.
	rightMargin	Right margin in pixels.
	topMargin	Top margin in pixels.
	bottomMargin	Bottom margin in pixels.
Returns	Returns 0 (success).	
Description	Shows the text in the parameter text on the display, existing text and/or image in the specified window is cleared. The text is shown on the display in accordance with the margins and the cnf parameter. See also section 10.4.5, Configuration of text and image. Text will automatically wrap to fit but carriage return can also be used to force line breaks.	

Examples	cnf = 1;
	text = "this is a text";
	result = gw.ShowTextL(cnf, text);
	
	or
	
	cnf = 1;
	text = "this is a text";
	result = gw.ShowTextL(cnf, text, 20, 0, 20, 0);	

10.8.5 ShowTextAndImage()

Show text and image on the display

Synopsis:	ShowTextAndImage(cnf, txtName, imageName)
	or
	ShowTextAndImage(cnf, txtName, imageName, leftMargin, rightMargin, topMargin, bottomMargin)

Arguments	cnf How to show the text. txtName The name of the text. imageName The name of the image. Optional (All or none) leftMargin Left margin in pixels. rightMargin Right margin in pixels. topMargin Top margin in pixels. bottomMargin Bottom margin in pixels.
Returns	Returns 0 for success and –1 for error.
Description	Shows the text identified by txtName on top of the image identified by imageName on the display, existing text and/or image in the specified window is cleared. The text is shown on the display in accordance with the margins and the cnf parameter. See also section 10.4.5, Configuration of text and image. Text will automatically wrap to fit but carriage return can also be used to force line breaks. The image fills the selected window independent of the original size and aspect ratio of the images. Margins only affect the text.

Example	cnf = 1;
	txtName = "name";
	imageName = "image";
	result = gw.ShowTextAndImage(cnf, txtName, imageName);
	
	or
	
	cnf = 1;
	txtName = "name";
	imageName = "image";
	res = gw.ShowTextAndImage(cnf, txtName, imageName, 20,0 20,0);

Note	The name of the text = Voicefilename (name of the file that contains text and/or sound)

10.8.6 ShowTextAndImageL()

Show literal text and image on the display

Synopsis:	ShowTextAndImageL(cnf, text, imageName)
	or
	ShowTextAndImageL(cnf, text, imageName, leftMargin, rightMargin, topMargin, bottomMargin)

Arguments	cnf How to show the text. text The text to show. imageName The name of the image. Optional (All or none) leftMargin Left margin in pixels. rightMargin Right margin in pixels. topMargin Top margin in pixels. bottomMargin Bottom margin in pixels.
Returns	Returns 0 for success and –1 for error.
Description	Shows the text in the parameter text on top of the image identified by imageName on the display, existing text and/or image in the specified window is cleared. The text is shown on the display in accordance with the margins and the cnf parameter. See also section 10.4.5, Configuration of text and image. Text will automatically wrap to fit but carriage return can also be used to force line breaks. The image fills the selected window independent of the original size and aspect ratio of the images. Margins only affect the text.

Example	cnf = 1;
	text = "this is a text";
	imageName = "image";
	result = gw.ShowTextAndImageL(cnf, text, imageName);
	
	or
	
	cnf = 1;
	text = "this is a text";
	imageName = "image";
	res = gw.ShowTextAndImageL(cnf, text, imageName, 20, 0, 20, 0);

10.8.7 ShowImage()

Show image on the display

Synopsis:	ShowImage(cnf, imageName)
Arguments	cnf How to show the text. imageName The name of the image.
Returns	Returns 0 for success and –1 for error.
Description	Shows the image identified by imageName on the display, existing text and/or image in the specified window is cleared. The image fills the selected window independent of the original size and aspect ratio of the images. See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	imageName = "image";
	result = gw.ShowImage(cnf, imageName);

Note	The name of the text = Voicefilename (name of the file that contains

	text and/or sound)

10.8.8 ClrText()

Clear text on the display

Synopsis:	ClrText(int cnf)
Arguments	cnf The text position to clear.
Returns	Returns 0 (success).
Description	Clears the display in accordance with the cnf parameter. See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	result = gw.ClrText(cnf);

10.8.9 ShowMenu()

Show menu on the display

Synopsis:	ShowMenu(cnf, select, menu)
Arguments	cnf How to show the menu. select Entry number of the entry to initially show as selected. menuAn array with identification of the menu entries (texts). Identification of the entries is given as names.
Returns	Returns 0 for success and –1 for error.
Description	A menu with text lines, identified by the array entries, is shown on the display. If configured as "Menu" by the "cnf" argument, The user can scroll between the entries and select one of them. The entries are implicitly numbered 0, 1, … This numbering is used to identify the user choice in return events (see section 10.8.17, WaitForEvent). If configured as "Info" by the "cnf" argument, The user can only scroll between the entry lines and no position marking is used. The text position *Menu header window* is intended for displaying a menu header. The text must be explicitly shown using the ShowText() or ShowTextL() commands if needed. See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	select = 0;
	menu = new Array();
	menu[0] = "entry0Name";
	menu[1] = "entry1Name";
	menu[2] = "entry2Name";
	result = gw.ShowMenu(cnf, select, menu);

10.8.10 ShowMenuL()

Show literal menu on the display

Synopsis:	ShowMenuL(cnf, select, menu)
Arguments	cnf How to show the menu. select Entry number of the entry to initially show as selected. menu An array with menu entries (texts).
Returns	Returns 0 for success and –1 for error.
Description	A menu with text lines, identified by the array entries, are shown on the display.

	If configured as “Menu” by the “cnf” argument, The user can scroll between the entries and select one of them. The entries are implicitly numbered 0, 1, … This numbering is used to identify the user choice in return events (see section 10.8.17, WaitForEvent). If configured as “Info” by the “cnf” argument, The user can only scroll between the entry lines and no position marking is used. The text position *Menu header window* is intended for displaying a menu header. The text must be explicitly shown using the ShowText() or ShowTextL() commands if needed. See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	select = 0;
	menu = new Array();
	menu[0] = “This is entry number 0”;
	menu[1] = “This is entry number 1”;
	menu[2] = “This is entry number 2”;
	result = gw.ShowMenuL(cnf, select, menu);

10.8.11 ClrMenu()

Clear menu on the display

Synopsis:	ClrMenu()
Arguments	None.
Returns	Returns 0 (success).
Description	Clear the menu on the display.
Example	result = gw.ClrMenu();

10.8.12 ShowButtons()

Show buttons on the display

Synopsis:	ShowButtons(cnf, button)
Arguments	cnf How to show the buttons. button An array with identification of the button entries (texts). Identification of the entries is given as names. The first entry is for the left button and the second entry is for the right button. The scroll buttons and the Info button are not labeled. Either of the entries may be an empty string.
Returns	Returns 0 if success. Returns –1 if error.
Description	Shows soft buttons on the display. (see section 10.8.17, WaitForEvent). Please note that for the buttons to start accepting key presses they must be explicitly activated using ActivateButtons(). See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	button = new Array();
	button[0] = “YESName”;
	button[1] = “NOName”;
	result = gw.ShowButtons(cnf, button);

10.8.13 ShowButtonsL()

Show literal buttons on the display

Synopsis:	ShowButtonsL(cnf, button)

Arguments	cnf How to show the buttons. button An array with the button entries (texts). The first entry is for the left button and the second entry is for the right button. The scroll buttons and the Info button are not labeled. Either of the entries may be an empty string.
Returns	Returns 0 if success. Returns −1 if error.
Description	Shows soft buttons on the display. (see section 10.8.17, WaitForEvent). Please note that for the buttons to start accepting key presses they must be explicitly activated using ActivateButtons(). See also section 10.4.5, Configuration of text and image.

Example	cnf = 1;
	button = new Array();
	button[0] = "YES";
	button[1] = "NO";
	result = gw.ShowButtonsL(cnf, button);

10.8.14 ClrButtons()

Clear buttons on the display

Synopsis:	ClrButtons()
Arguments	None.
Returns	Returns 0 (success).
Description	Soft buttons and handling of soft buttons is cleared.
Example	result = gw.ClrButtons();

10.8.15 ActivateButtons()

Activate buttons on the display

Synopsis:	ActivateButtons(act)
Arguments	act Binary pattern where bits identifies a given button. 0 ⇔ do not activate, 1 ⇔ activate. The (physical) buttons are identified as follows: bit 0 : LEFT button, bit 1: RIGHT button, bit 2: SCROLL UP button, bit 3: SCROLL DOWN button and and bit 4 INFO button.
Returns	Returns 0 (success).
Description	The buttons given by *act* are activated i.e. user responses on these buttons are recognized. When an activated button gets pressed all buttons are immediately deactivated automatically, and must be activated again before they can be used again.

Example	act = 3;
	result = gw.ActivateButtons(act);

10.8.16 PlaySound()

Voice a phrase

Synopsis:	PlaySound(cnt, voiceName)
Arguments	cnt Continuation indicator. If 0 then this is part of a combined phrase and more will follow. If 1 then this is the last part of a combined part or the only part of a not combined phrase. voiceName The name of a phrase identifying a sound file.
Returns	Returns 0 if success. Returns −1 if error.
Description	Voice the sound file given by voiceName. Multiple phrases may be

	queued by calling PlaySound() successively. The last call to PlaySound() must have the *cnt* argument set. A PHRASE event (see section 10.8.17, WaitForEvent()) is generated to indicate when the last phrase has been spoken. After having called PlaySound() with the *cnt* argument set, PlaySound() may not be called again until the PHRASE event has been received.

Example	cnt = 0;
	voiceName = "soundName_1";
	result = gw.PlaySound(cnt, voiceName);
	cnt = 1;
	voiceName = "soundName_2";
	result = gw.PlaySound(cnt, voiceName);

Note	A maximum of 25 sound files can be used in one combined phrase.

10.8.17 WaitForEvent()

Wait for an event

Synopsis:	WaitForEvent(event)	
Arguments	event	For a SCROLL event the first entry of the returned array indicates the choice selected by the user in the menu or -1 if no menu is active (See ShowMenu) and the second entry in the array specifies which SCROLL key was pressed 0 for DOWN and 1 for UP." For a timeout event the first entry of the returned array identifies the timeout type (see section 10.8.33, SetTimeout). For a HOST event the first entry of the returned array identifies the type of the telegram (<type> tag) sent to the internet server. 1. REQUESTREGISTRATION 2. REQUESTFACTORYTEST 3. REQUESTPATIENTINITIALIZATION 4. FORCECONNECT And the second entry contains a return code: 0 if the message was delivered successfully -1 otherwise. For a PACKAGE event the first entry of the returned array contains the package number transmitted and the second entry contains the remaining number of messages in the queue. For DISCONNECT event the first entry of the returned array contains the status: 0 for success otherwise -1. And the second entry contains the number of transmitted packages.
Returns	Identifies which type of event happened. Event identifications are given by the gw.—Event() functions (see later). Returns –1 in case of error.	
Description	Wait for an event to happen. There are a number of possible events, which can happen. We can wait for the user to press a button or we can wait for a timeout to elapse. There are also other possibilities (see gw.—Event()).	

Example	event = new Array();
	type = gw.WaitForEvent(event);

10.8.18 ClrResp()

Clear user responses

Synopsis:	ClrResp()
Arguments	None
Returns	Returns 0 (success).
Description	The user response memory arrear is cleared. This does not affect any messages already queued for sending.
Example	result = gw.ClrResp()

10.8.19 AddToResp()

Add to user responses

Synopsis:	AddToResp(response)	
Arguments	response	String to add to the information to be sent to the Internet server. NOTE that the characters "{" and "}" are not allowed.
Returns	Returns 0 if success. Returns −1 if error.	
Description	A string is added to the User response memory arrear that holds the information to be sent to the Internet server.	

Example	response = "message to Internet Server.\n";
	result = gw.AddToResp(response);	

10.8.20 SendRespToServer()

Send responses to server

Synopsis:	SendRespToServer()
Arguments	None.
Returns	Returns 0 for success and −1 for error.
Description	The collected information (by AddResp()) in the user response memory arrear is queued for sending to the Internet server. A HOST event (see section 10.8.17, WaitForEvent()) is NOT generated to indicate success or failure in contacting the Internet server.
Example	Result = gw.SendRespToServer();

10.8.21 GetLastMeasurement()

Get the last measurement value

Synopsis:	GetLastMeasurement()
Arguments	None.
Returns	A string containing the DeviceId and Value fields of the last measurement telegram sent to the server.
Description	The content of <DeviceId> and <Value> tags in the last XML

	telegram with measurement data. For details about the format please see section 12.2.1, Value formats.
Example	Result = gw.GetLastMeasurement();

10.8.22 GetLastResp()

Get the last response

Synopsis:	GetLastResp()
Arguments	None.
Returns	A string containing the DeviceId and Value fields of the last user response telegram sent to the server.
Description	The content of <DeviceId> and <Value> tags in the user response telegram that was sent in response to the last measurement data. For details about the format please see chapter 12, XML.
Example	Result = gw.GetLastResp();

10.8.23 GetTime()

Get time

Synopsis:	GetTime()
Arguments	None.
Returns	The time (number of seconds since 1-jan-1970).
Description	Returns the RTX337x time.
Example	time = gw.GetTime();
Note	The Date class from the core JavaScript language does not support getting the current time from eCos. This method must therefore be used when the current time is needed.

10.8.24 SetTime()

Set time

Synopsis:	SetTime(time)	
Arguments	time	The time (number of seconds since 1-jan-1970).
Returns	Returns 0 (success).	
Description	Sets the RTX337x time.	

Example	time = 1252667800;
	gw.SetTime(time);	

Note	The Date class from the core JavaScript language does not support setting the current time from eCos. This method must therefore be used when setting of the current time is needed.	

10.8.25 EnableConfiguration()

Enable tty configuration

Synopsis:	EnableConfiguration(int seconds)	
Arguments	seconds	Timeout (in seconds) for tty to be enabled. If "seconds" is zero then no timeout is set.
Returns	Returns 0 (success).	

Description	Enable the possibility for configuring the RTX337x from the tty interface. There is a "seconds" inactivity timeout on this.

Example	seconds = 15 * 60;
	result = gw.EnableConfiguration(seconds);

10.8.26 DisableConfiguration()

Disable tty configuration

Synopsis:	DisableConfiguration()
Arguments	None.
Returns	Returns 0 (success).
Description	Disable the possibility for configuring the RTX337x from the tty interface.
Example	result = gw.DisableConfiguration();

10.8.27 Reset()

Reset RTX337x

Synopsis:	Reset()
Arguments	None.
Returns	Returns 0 (success).
Description	Resets the RTX337x application.
Example	result = gw.Reset();

10.8.28 GetScriptName()

Get current JavaScript name

Synopsis:	GetScriptName()
Arguments	None.
Returns	The name (identification) of the current JavaScript. Returns empty string in case of error.
Description	Gets the identification of the current JavaScript.
Example	scriptName = gw.GetScriptName();

10.8.29 GetActivatorId()

Get activator identification

Synopsis:	GetActivatorId()
Arguments	None.
Returns	The identification of the activator of the JavaScript (e.g. weight device). Returns empty string in case of error.
Description	Gets the identification of the activator of the JavaScript.
Example	scriptName = gw.GetActivatorId();

10.8.30 GetActivatorType()

Get activator type

Synopsis:	GetActivatorType()
Arguments	None.
Returns	The type of the activator of the JavaScript (e.g. ADBTWSType). For device activated scripts, this is the device type. For other activations the following strings are returned: • Key combination: “KeyComb” • Script function ActivateScript: “Script” • AT-command AT+pACTIVATE “AT” • Default Script “Default” Returns empty string in case of error.
Description	Gets the type of the activator of the JavaScript.
Example	TypeName = gw.GetActivatorType();

10.8.31 GetNrScriptPars()

Get number of parameters

Synopsis:	GetNrScriptPars()
Arguments	None.
Returns	Number of parameters this JavaScript was activated with.
Description	Gets the number of parameters this JavaScript was activated with.
Example	nrScriptPars = gw.GetNrScriptPars();

10.8.32 GetScriptPar()

Get a parameter

Synopsis:	GetScriptPar(index)	
Arguments	index	The index of the parameter. There can be a maximum of 25 parameters for a JavaScript. The parameters are organized as a double array with index starting from 0. See parameter map for specific devices in chapter 13, Installation of External Devices.
Returns	The given parameter from the double array. The meaning of these parameters depends on the JavaScript activator. For a weight device this might e.g. be the weight.	
Description	Gets a given parameter.	

Example	param = new Array();
	nrScriptPars = gw.GetNrScriptPars();
	for (i = 0; i < nrScriptPars; i++)
	    param[i] = gw.GetScriptPar(i);	

10.8.33 SetTimeout()

Set timeout

Synopsis:	SetTimeout(timeoutType, delay)
Arguments	timeoutType Identification of the timeout. (0 <= timeoutType < 2^30). delay The timeout in seconds. (0 < delay < 36000)
Returns	Returns 0 if success. Returns −1 if error.
Description	Set timeout with a given timeout type. More than one timeout (with different timeout types) can be set at the same time. A maximum of 20 timeouts can set at the time. When the timeout expires a TIMEOUT event is generated (see section 10.8.17, WaitForEvent()).

Example	timeoutType = 10;
	delay = 15;
	result = gw.SetTimeout(timeoutType, delay);

Note:	The precision of the timeout is within 1 second.

10.8.34 ClrTimeout()

Clear timeout

Synopsis:	ClrTimeout(timeoutType)
Arguments	timeoutType Identification of the timeout. (0 <= timeoutType < 2^30).
Returns	Returns 0 if success. Returns −1 if error.
Description	Clears timeout for a given timeout type.

Example	timeoutType = 10;
	result = gw.ClrTimeout(timeoutType);

10.8.35 ClrAllTimeouts()

Clear all timeouts

Synopsis:	ClrAllTimeouts()
Arguments	None
Returns	Returns 0 if success. Returns −1 if error.
Description	Clears all active timeouts.
Example	result = gw.ClrAllTimeouts();

10.8.36 LEvent()

LEFT event type

Synopsis:	LEvent()
Arguments	None
Returns	Returns the LEFT event type
Description	Return the type for when user has pressed the LEFT soft button.
Example	type = gw.LEvent();

10.8.37 REvent()

RIGHT event type

Synopsis:	REvent()
Arguments	None
Returns	Returns the RIGHT event type
Description	Return the type for when user has pressed the RIGHT soft button.
Example	type = gw.REvent();

10.8.38 SEvent()

SCROLL event type

Synopsis:	SEvent()
Arguments	None
Returns	Returns the SCROLL event type.
Description	Return the type for when user has pressed one of the SCROLL soft buttons.
Example	type = gw.SEvent();

10.8.39 IEvent()

INFO event type

Synopsis:	IEvent()
Arguments	None
Returns	Returns the INFO event type.
Description	Return the type for when user has pressed the INFO button.
Example	type = gw.IEvent();

10.8.40 PHRASEEvent()

PHRASE event type

Synopsis:	PHRASEEvent()
Arguments	None
Returns	Returns the PHRASE event type (type for when a phrase has been spoken).
Example	type = gw.PHRASEEvent();

10.8.41 HOSTEvent()

HOST event type

Synopsis:	HOSTEvent()
Arguments	None
Returns	Returns the HOST event type (type for when a JavaScript generated telegram has been sent to the Host (internet) Server.
Example	type = gw.HOSTEvent();

10.8.42 DIALEvent()

DIAL event type

Synopsis:	DIALEvent()
Arguments	None
Returns	Returns the DIAL event type (type for when a JavaScript generated server connection starts to dial.
Example	type = gw.DIALEvent();
Note	This event is only generated if gw.Connect() has been called from the same script and returned > 0 or -2.

10.8.43 CONNECTEvent()

CONNECT event type

Synopsis:	CONNECTEvent()
Arguments	None
Returns	Returns the CONNECT event type (type for when a JavaScript generated server connection is connected to the ISP).
Example	type = gw.CONNECTEvent();
Note	This event is only generated if gw.Connect() has been called from the same script and returned > 0 or -2.

10.8.44 DISCONNECTEvent()

DISCONNECT event type

Synopsis:	DISCONNECTEvent()
Arguments	None
Returns	Returns the DISCONNECT event type (type for when a JavaScript generated server connection is finished and the modem has disconnected from the ISP).
Example	type = gw.DISCONNECTEvent();
Note	This event is only generated if gw.Connect() has been called from the same script and returned > 0 or -2.

10.8.45 PACKAGEEvent()

PACKAGE event type

Synopsis:	PACKAGEEvent()
Arguments	None
Returns	Returns the PACKAGE event type (type for when a Javacript generated server connection has successfully transmitted a package to the host server).
Example	type = gw.PACKAGEEvent();
Note	This event is only generated if gw.Connect() has been called from the same script and returned > 0 or -2. The time to send a package includes the time for downloading data from the server.

10.8.46 TIMEOUTEvent()

TIMEOUT event type

Synopsis:	TIMEOUTEvent()
Arguments	None
Returns	Returns the TIMEOUT event type (type for when a timeout has elapsed – see section 10.8.33, gw.SetTimeout()).
Example	type = gw.TIMEOUTEvent();

10.8.47 EXITEvent()

EXIT event type

Synopsis:	EXITEvent()
Arguments	None
Returns	Returns the EXIT event type (type for when JavaScript is requested to exit).
Example	type = gw.EXITEvent();

10.8.48 StopSystem()

Stop the system

Synopsis:	StopSystem()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Request to stop the system for detecting modem parameters. Notice stopping the system might take some time.
Example	result = gw.StopSystem();
Note	Care should be taken to always start the system again when it has been stopped. The system should never be stopped from both sources (TTY and JavaScript) at the same time. If the TTY stopped the system it should start it again before it is stopped from a JavaScript. This process might take a while, as it won't return while the system is sending messages to the Host Server.

10.8.49 StartSystem()

Start the system

Synopsis:	StartSystem()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Request to start the system. The system must be started after having been stopped for detecting modem control parameters.
Example	result = gw.StartSystem();
Note	The system should never be started from another source (TTY and JavaScript) than the one it was stopped from. If the TTY stopped the system it should start it again before it is stopped from a JavaScript.

10.8.50 GetMBeg1()

Get MBEG1 modem initialization string

Synopsis:	GetMBeg1(id)
Arguments	id Identification of the modem initialization string. The id is an index into the MBEG1 vector (starting from 0). See description of the AT-command: "AT+pMBEG1" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem initialization string. Returns empty string in case of error.
Description	Gets the given modem initialization string from the MBEG1 vector of strings.

Example	id = 1;
	MBeg1 = gw.GetMBeg1(id);

Note	Details about setting the vector referred to above can be found in section 9.2.5, AT+pMBEG1_VC.

10.8.51 GetMBeg2()

Get MBEG2 modem initialization string

Synopsis:	GetMBeg2(id)
Arguments	id Identification of the modem initialization string. The id is an index into the MBEG2 vector (starting from 0). See description of the AT-command: "AT+pMBEG2" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem initialization string. Returns empty string in case of error.
Description	Gets the given modem initialization string from the MBEG2 vector of strings.

Example	id = 1;
	MBeg2 = gw.MGetBeg2(id);

Note	Details about setting the vector referred to above can be found in section 9.2.7, AT+pMBEG2_VC.

10.8.52 GetMPreD1()

Get MPRED1 modem pre dial string

Synopsis:	GetMPreD1(id)

Arguments	id Identification of the modem pre dial string. The id is an index into the MPRED1 vector (starting from 0). See description of the AT-command: "AT+pMPRED1" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem pre dial string. Returns empty string in case of error.
Description	Gets the given modem pre dial string from the MPRED1 vector of strings.

Example	id = 1;
	MPreD1 = gw.GetMPreD1(id);

Note	Details about setting the vector referred to above can be found in section 9.2.11, AT+pMPRED1_VC.

10.8.53 GetMPreD2()

Get MPRED2 modem pre dial string

Synopsis:	GetMPreD2(id)
Arguments	id Identification of the modem pre dial string. The id is an index into the MPRED2 vector (starting from 0). See description of the AT-command: "AT+pMPRED2" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem pre dial string. Returns empty string in case of error.
Description	Gets the given modem pre dial string from the MPRED2 vector of strings.

Example	id = 1;
	MpreD2 = gw.GetMPreD2(id);

Note	Details about setting the vector referred to above can be found in section 9.2.13, AT+pMPRED2_VC.

10.8.54 GetMPreD3()

Get MPRED3 modem pre dial string

Synopsis:	GetMPreD3(id)
Arguments	id Identification of the modem pre dial string. The id is an index into the MPRED3 vector (starting from 0). See description of the AT-command: "AT+pMPRED3" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem pre dial string. Returns empty string in case of error.
Description	Gets the given modem pre dial string from the MPRED3vector of strings.

Example	id = 1;
	MpreD3= gw.GetMPreD3id);

Note	Details about setting the vector referred to above can be found in section 9.2.15, AT+pMPRED3_VC.

10.8.55 GetMEnd()

Get MEND modem clean up string

Synopsis:	GetMEnd(id)
Arguments	id Identification of the modem clean up string. The id is an index into the MEND vector (starting from 0). See description of the AT-command: "AT+pMEND" in Chapter 9, AT+p Command Set, for details.
Returns	Returns the modem clean up string. Returns empty string in case of error.
Description	Gets the given modem clean up string from the MEND vector of strings.

Example	id = 1;
	MEnd = gw.GetMEnd(id);

Note	Details about setting the vector referred to above can be found in section 9.2.17, AT+pMEND_VC.

10.8.56 GetMCDBeg()

Get MCDBEG modem control dial string

Synopsis:	GetMCDBeg(id)
Arguments	id Identification of the modem control dial string. The id is an index into the MCDBEG vector (starting from 0). See section 9.2.8, AT+pMCDBEG, for details.
Returns	Returns the modem control dial string. Returns empty string in case of error.
Description	Gets the given modem control dial string from the MCDBEG vector of strings.

Example	id = 1;
	MCDBeg = gw.GetMCDBeg(id);

Note	Details about setting the vector referred to above can be found in section 9.2.9, AT+pMCDBEG_VC.

10.8.57 GetMCDEnd()

Get MCDEND modem control dial string

Synopsis:	GetMCDEnd(id)
Arguments	id Identification of the modem control dial string. The id is an index into the MCDEND vector (starting from 0). See section 9.2.18, AT+pMCDEND, for details.
Returns	Returns the modem control dial string. Returns empty string in case of error.
Description	Gets the given modem control dial string from the MCDEND vector of strings.

Example	id = 1;
	MCDEnd = gw.GetMCDEnd(id);

Note	Details about setting the vector referred to above can be found in section 9.2.19, AT+pMCDEND_VC.

10.8.58 GetISP()

Get Internet Service Provider (ISP) phone number

Synopsis:	GetISP(id)
Arguments	id Identification of the phone number string. The id is an index into the ISP vector (starting from 0). See section 9.2.1, AT+pISP, for details.
Returns	Returns the Internet Service Provider phone number string. Returns empty string in case of error.
Description	Gets the Internet Service Provider phone number string from the ISP vector of strings.

Example	id = 1;
	ISP = gw.GetISP(id);

Note	Details about setting the vector referred to above can be found in section 9.2.3, AT+pISP_VC.

10.8.59 SetMBeg1()

Set MBEG1 modem initialization string

Synopsis:	SetMBeg1(MBeg1)
Arguments	MBeg1 The MBEG1 modem initialization string. See section 9.2.4, AT+pMBEG1, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Sets the given modem initialization string as the MBEG1 string in the RTX337x database.

Example	id = 1;
	MBeg1 = gw.GetMBeg1(id);
	result = gw.SetMBeg1(MBeg1);

10.8.60 SetMBeg2()

Set MBEG2 modem initialization string

Synopsis:	SetMBeg2(MBeg2)
Arguments	MBeg2 The MBEG2 modem initialization string. See section 9.2.6, AT+pMBEG2, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Sets the given modem initialization string as the MBEG2 string in the RTX337x database.

Example	id = 1;
	MBeg2 = gw.GetMBeg2(id);
	result = gw.SetMBeg2(MBeg2);

10.8.61 SetMPreD1()

Set MPRED1 modem pre dial string

Synopsis:	SetMPreD1(MPreD1)	
Arguments	MPreD1	The MPRED1 modem pre dial string. See section 9.2.10, AT+pMPRED1, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem pre dial string as the MPRED1 string in the RTX337x database.	

Example	id = 1;
	MPreD1 = gw.GetMPreD1(id);
	result = gw.SetMPreD1(MPreD1);	

10.8.62 SetMPreD2()

Set MPRED2 modem pre dial string

Synopsis:	SetMPreD2(MpreD2)	
Arguments	MpreD2	The MPRED2 modem pre dial string. See section 9.2.12, AT+pMPRED2, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem pre dial string as the MPRED2 string in the RTX337x database.	

Example	id = 1;
	MPreD2 = gw.GetMPreD2(id);
	result = gw.SetMPreD2(MPreD2);	

10.8.63 SetMPreD3()

Set MPRED3 modem pre dial string

Synopsis:	SetMPreD3(MpreD3)	
Arguments	MpreD3	The MPRED3 modem pre dial string. See section 9.2.14, AT+pMPRED3, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem pre dial string as the MPRED3 string in the RTX337x database.	

Example	id = 1;
	MPreD3 = gw.GetMPreD3(id);
	result = gw.SetMPreD3(MPreD3);	

10.8.64 SetMEnd()

Set MEND modem clean up string

Synopsis:	SetMEnd(MEnd)	
Arguments	MEnd	The MEND modem clean up string. See section 9.2.16, AT+pMEND, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem clean up string as the MEND string in the RTX337x database.	

Example	id = 1;
	MEnd = gw.GetMEnd(id);
	result = gw.SetMEnd(MEnd);	

10.8.65 SetMCDBeg()

Set MCDBEG modem control dial string

Synopsis:	SetMCDBeg(MCDBeg)	
Arguments	MCDBeg	The MCDBeg modem control dial string. See section 9.2.8, AT+pMCDBEG, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem control dial string as the MCDBEG string in the RTX337x database.	

Example	id = 1;
	MCDBeg = gw.GetMCDBeg(id);
	result = gw.SetMCDBeg(MCDBeg);	

10.8.66 SetMCDEnd()

Set MCDEND modem control dial string

Synopsis:	SetMCDEnd(MCDEnd)	
Arguments	MCDEnd	The MCDEnd modem control dial string. See section 9.2.18, AT+pMCDEND, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given modem control dial string as the MCDEND string in the RTX337x database.	

Example	id = 1;
	MCDEnd = gw.GetMCDEnd(id);
	result = gw.SetMCDEnd(MCDEnd);	

10.8.67 SetISP()

Set Internet Service Provider (ISP) phone number string

Synopsis:	SetISP(ISP)	
Arguments	ISP	The Internet Service Provider phone number string. See section 9.2.1, AT+pISP for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given Internet Service Provider phone number string as the ISP string in the RTX337x database.	

Example	id = 1;
	ISP = gw.GetISP(id);
	result = gw.SetISP(ISP);	

10.8.68 SetISPBck()

Set Internet Service Provider (ISP) backup phone number string

Synopsis:	SetISPBck(ISP)	
Arguments	ISP	The Internet Service Provider backup phone number string. See section 9.2.2, AT+pISP_BACKUP, for details.
Returns	Returns 0 if success. Returns –1 if error.	
Description	Sets the given Internet Service Provider backup phone number string as the ISP_BACKUP string in the RTX337x database.	

Example	id = 1;
	ISP = gw.GetISP(id);
	result = gw.SetISPBck(ISP);	

10.8.69 ResetModem()

Resets the modem

Synopsis:	ResetModem()
Arguments	None
Returns	Returns 0 if success. Returns –1 if error.
Description	Resets the modem. The system must be stopped to use this command.
Example	result = gw.ResetModem();

10.8.70 ATModemCommand()

Execute AT command on the modem

Synopsis:	ATModemCommand(command, timeout)	
Arguments	command timeout	A string with the modem command. For RTX3370 see the document: "an93_09.pdf" from Silicon Laboratories for details. For the RTX3371 see the document: "Telit_AT_Commands_Reference_Guide_r5.pdf" from Telit for details. Max. time in seconds to wait for response from modem
Returns	The response to the AT command from the modem. Returns empty string in case of error.	
Description	Executes a modem AT command and returns the result. The system must be stopped to use this command. The command string is sent unedited to the modem (no CR appended).	

Example	command = "ATZ\r";
	timeout = 2;
	result = gw.ATModemCommand(command, timeout);	

10.8.71 GetConfigFromHost()

Get configuration from Host Server

Synopsis:	GetConfigFromHost()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Request RTX337x configuration from the Host (internet) Server. A HOST event (see section 10.8.17, WaitForEvent) is generated to indicate success or failure in contacting the Internet server. This function uses the "REQUESTPATIENTINITIALIZATION" telegram. See Chapter 12, XML for details.
Example	result = gw.GetConfigFromHost();

10.8.72 RegisterOnHost()

Register RTX337x on Host Server

Synopsis:	RegisterOnHost()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Request registration on the Host (internet) Server. A HOST event (see section 10.8.17, WaitForEvent) is generated to indicate success or failure in contacting the Internet server. This function uses the "REQUESTREGISTRATION" telegram. See Chapter 12, XML for details.
Example	result = gw.RegisterOnHost();

10.8.73 GetTestFromHost()

Get test from Host Server

Synopsis:	GetTestFromHost()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Request test JavaScript + activation of test JavaScript from the Host (internet) Server. A HOST event (see section 10.8.17, WaitForEvent) is generate to indicated success or failure in contacting the Internet server. This function uses the "REQUESTFACTORYTEST" telegram. See Chapter 12, XML for details.
Example	result = gw.GetTestFromHost();

10.8.74 ForceConnect()

Force a connection to the Host Server

Synopsis:	ForceConnect()
Arguments	None
Returns	Returns 0 for success and –1 for error.
Description	Forces a connection to the Host (Internet) Server. A HOST event (see section 10.8.17, WaitForEvent) is generated to indicate success or failure in contacting the Internet server. This function uses the "FORCECONNECT" telegram. See Chapter 12, XML for details.
Example	result = gw.ForceConnect();

10.8.75 Connect()

Initialize a connection to the host server.

Synopsis:	Connect()
Arguments	None
Returns	Returns number of messages currently queued for transmission, 0 if there are no messages queued for transmission (this will not trigger a connection attempt), -1 for error and -2 if dial is already in progress.
Description	Initializes a connection to the host (Internet server). Events are generated (if return value is positive or -2) as the connection progresses see DIALEvent(), CONNECTEvent(), DISCONNECTEvent() and PACKAGEEvent().
Example	result = gw.Connect();
Note	Number of transmitted packages might be larger then the returned value if new data is queued for transmission during the connection. If the call is attempted while a server connection is already in progress -2 will be returned. If AT+pAUTODIAL is set to 0 then this will be the ONLY condition other than error flag changes that causes a connection attempt.

10.8.76 GetHostConnectionStatus()

Get status for latest Host Server connection

Synopsis:	GetHostConnectionStatus()
Arguments	None
Returns	Returns a text.
Description	The returned text describes the status for Host Server connection. The text consists of digits only. See description of the AT-command: "AT+pHOSTSTATUS" in Chapter 9, AT+p Command Set, for details.
Example	hostStatusText = gw.GetHostConnectionStatus();

10.8.77 GetStatus()

Get RTX337x status

Synopsis:	GetStatus()
Arguments	None
Returns	Returns a text.
Description	The returned text describes the status for the RTX337x. The text consists of digits only. See section 9.4.17, AT+pSTATUS, for details.
Example	Status = gw.GetStatus();

10.8.78 GetCurrentPeripherals()

Get identification of current peripherals

Synopsis:	GetCurrentPeripherals(peripherals)	
Arguments	Peripherals	The entries of the returned array are text strings describing the current peripherals. This is the device name used by the AT-command AT+pINSDV.
Returns	Returns 0 for success and –1 for error	
Description	Gets a description of the current peripherals.	

Example	peripherals = new Array();
	result = gw.GetCurrentPeripherals(peripherals);	

10.8.79 ActivateScript()

Activate JavaScript

Synopsis:	ActivateScript(scriptName, priority, scriptPars, noPar, activateId)	
Arguments	scriptName	The name of the JavaScript to activate. This is a string.
	priority	The priority of the JavaScript to activate. 0 is the lowest priority.
	scriptPars	Parameters for the JavaScript. The parameters are in a double array.
	noPar	Number of parameters (max. is 25).
	activateId	Identification of the activator. This is a string.
Returns	Returns 0 for success and –1 for error	
Description	Request activation of JavaScript, when the current JavaScript exits.	
Example	scriptPar = new Array();	

	scriptName = “system”;
	priority = 10;
	scriptPar[0] = 1.1;
	scriptPar[1] =0.0;
	noPar = 2;
	activateId = “me”;
	result = gw.ActivateScript(scriptName, priority, scriptPars, noPar, activateId);

10.8.80 Log()

Log a message

Synopsis:	Log(text)
Arguments	text The text to log. This is a string.
Returns	Returns 0 for success and –1 for error
Description	Writes an entry in the log. The log type is: “LOG_NET, LOG_INFO”.
Example	text= “This is a message to be logged in the log”; result = gw.Log(text);
Note	Log entries are limited to 511 characters each.

10.8.81 SetFACTORYMode()

Set FACTORY mode

Synopsis:	SetFACTORYMode()
Arguments	None.
Returns	Always returns 0 (success).
Description	Sets the FACTORY mode of the RTX337x system. See section 10.10.1, RTX337x Modes for details.
Example	result = gw.SetFACTORYMode();

10.8.82 SetPATIENTMode()

Set PATIENT mode

Synopsis:	SetPATIENTMode()
Arguments	None.
Returns	Always returns 0 (success).
Description	Sets the PATIENT mode of the RTX337x system. See section 10.10.1, RTX337x Modes for details.
Example	result = gw.SetPATIENTMode();

10.8.83 SetNORMALMode()

Set NORMAL mode

Synopsis:	SetNORMALMode()
Arguments	None.
Returns	Always returns 0 (success).
Description	Sets the NORMAL mode of the RTX337x system. See section 10.10.1, RTX337x Modes for details.
Example	result = gw.SetNORMALMode();

10.8.84 SetBacklight()

Set display backlight mode

Synopsis:	SetBacklight()
Arguments	onoff 0=turn backlight off, 1=turn backlight on (dimmed), 2=turn backlight on ontime The number of seconds to keep backlight on.
Returns	Always returns 0 (success).
Description	Set the display backlight. If argument onoff is 2 the ontime argument controls how long time the backlight will be on before being automatically turned back to dimmed. If ontime is 0 the backlight is not dimmed automatically.
Example	result = gw.SetBacklight(1, 120);

10.8.85 GetSaveData()

Get JavaScript data.

Synopsis:	GetSaveData(name)
Arguments	name The piece of JavaScript data to retrieve (string ≤10)
Returns	Returns the string stored at the respective index or empty string on error (name not existing is considered an error).
Description	Retrieves the piece of data stored as name.
Example	result = gw.GetSaveData(store1);

10.8.86 SetSaveData()

Save JavaScript data.

Synopsis:	SetSaveData(name, data)
Arguments	name The name the data should be saved as (string ≤ 10) data The data to be saved.
Returns	Returns 0 if success, -2 if there is no space. Returns –1 if error.
Description	Stores the string in data in the JavaScript storage as name.
Example	data = "this is a string"; Result = gw.SetSaveData(store1, data);

10.8.87 InsertDevice()

Insert external device

Synopsis:	InsertDevice(string)
Arguments	String The insertion string. See description of the AT-command: "AT+pINSDV" in Chapter 9, AT+p Command Set, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Insert new device into the RTX337x.

Example	String="ADScl:ADBTWSType:FFFFFFFFFFFF-39121440-WS----90-60";
	result=gw.InsertDevice(string);

10.8.88 UpdateDevice()

Update configuration of inserted external device.

Synopsis:	UpdateDevice(string)
Arguments	String The update string. See description of the AT-command: "AT+pUPDDV" in Chapter 9, AT+p Command Set, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Update configuration of an inserted device on the RTX337x.

Example	String="ADScl:FFFFFFFFFFFF-39121440-WS----90-60";
	result=gw.UpdateDevice(string);

10.8.89 RemoveDevice()

Remove an inserted external device.

Synopsis:	RemoveDevice(name)
Arguments	name The device name.
Returns	Returns 0 if success. Returns –1 if error.
Description	Remove an inserted device on the RTX337x.
Example	Result = gw.RemoveDevice("ADScl");

10.8.90 SetDeviceTiming()

Set device timing.

Synopsis:	SetDeviceTiming(string)
Arguments	String The timing string. See description of the AT-command: "AT+pDV" in Chapter 9, AT+p Command Set, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Set timing for a specific device on the RTX337x.
Example	Result = gw.SetDeviceTiming("ADScl:I-0");

10.8.91 SetDeviceSupervision()

Set device supervision.

Synopsis:	SetDeviceSupervision (string)
Arguments	string The supervision string. See description of the AT-command: "AT+pDVS" in Chapter 9, AT+p Command Set, for details.
Returns	Returns 0 if success. Returns –1 if error.
Description	Set supervision for a specific device on the RTX337x.
Example	result = gw.SetDeviceSupervision("ADScl:I-0");

10.8.92 GetDeviceType()

Get inserted device type

Synopsis:	GetDeviceType(name)
Arguments	name The device name.
Returns	Returns a string containing the type name of the device or empty string on error (name not existing is considered an error). See type name for each device in chapter 13, Installation of External devices.
Description	Retrieve the type of a specific device.
Example	deviceType = gw.GetDeviceType("ADScl");

10.8.93 GetDeviceInfo()

Get inserted device info string

Synopsis:	GetDeviceInfo(name)
Arguments	name The device name.
Returns	Returns a string containing the info of the device or empty string on error (name not existing is considered an error). See info description for each device in chapter 13, Installation of External devices, for details.
Description	Retrieve the info of a specific device.
Example	deviceInfo = gw.GetDeviceInfo("ADScl");

10.8.94 GetDeviceTiming()

Get inserted device timing.

Synopsis:	GetDeviceTiming(name)
Arguments	name The device name.
Returns	Returns a string containing the timing of the device or empty string on error (name not existing is considered an error). See parameter description of AT-command: "AT+pDV" in Chapter 9, AT+p Command Set, for details.
Description	Retrieve the timing of a specific device.
Example	deviceTiming = gw.GetDeviceTiming("ADScl");

10.8.95 GetDeviceSupervision()

Get inserted device supervision.

Synopsis:	GetDeviceSupervision(name)
Arguments	name The device name.
Returns	Returns a string containing the supervision of the device or empty string on error (name not existing is considered an error). See parameter description of AT-command: "AT+pDVS" in Chapter 9, AT+p Command Set, for details.
Description	Retrieve the supervision of a specific device.
Example	deviceSupervision = gw.GetDeviceSupervision ("ADScl");

10.8.96 GetSignalQuality()

Get GSM signal quality and GPRS service state.

Synopsis:	GetSignalQuality(SignalQuality)
Arguments	SignalQuality[0] Signal Quality 0 = -113 dBm or less – MMI may indicate no antenna bar 1 – 5 = -111 – -103 dBm – MMI may indicate 1 antenna bar 6 – 9 = -101 – -95 dBm – MMI may indicate 2 antenna bar 10 – 14 = -93 – -85 dBm – MMI may indicate 3 antenna bar 15 – 31 = -83 dBm or more – MMI may indicate 4 antenna bar 99 = Not detected SignalQuality[1] Bit error rate (in percent) 0 = less than 0.2% 1 = 0.2% to 0.4% 2 = 0.4% to 0.8% 3 = 0.8% to 1.6% 4 = 1.6% to 3.2% 5 = 3.2% to 6.4% 6 = 6.4% to 12.8% 7 = more than 12.8% 99 = not known or not detectable SignalQuality[2] GPRS service state 0 = detached 1 = attached
Returns	0 = Success -1 = Error (GSM module not present is considered as an error.)
Description	Retrieve the GSM signal quality and GPRS service state.

Example	SignalQuality = new Array();
	result = gw. GetSignalQuality(SignalQuality);

10.8.97 GetModel()

Get Model name.

Synopsis:	GetModel()
Arguments	None.
Returns	Returns a string containing the model name: • “RTX3370” for the PSTN version. • “RTX3371” for the GSM/GPRS version.
Description	Return the model name.
Example	result = gw. GetModel();

10.8.98 GetVer()

Get Firmware version.

Synopsis:	GetVer()
Arguments	None.
Returns	Returns a string containing the Firmware version.
Description	Return the Firmware version.
Example	result = gw. GetVer();

10.9 JavaScript Scheduling

10.9.1 Scheduling of JavaScripts in the RTX337x

A Script Handler takes care of scheduling and execution of JavaScripts in the RTX337x. This Handler consists of one thread i.e. only one JavaScript can be executing at a time.

- Scripts are scheduled in the order the Script Handler receives them.
- Scripts have an assigned priority (default priority is 1).
- Scripts have waiting points where they can wait for events (button pressure, timeout, etc.). At waiting points the JavaScript is still assumed to be executing.
- In case no JavaScripts are currently executing, then a JavaScript requested to be scheduled will be started.
- In case a JavaScript, with the same or lower priority as the currently executing JavaScript, is requested to be scheduled, then it is queued internally in the Script Handler for later scheduling.
- In case a JavaScript, with a higher priority than the currently executing JavaScript, is requested to be scheduled then it is queued internally in the Script Handler for later scheduling. Simultaneously an “exit” event is generated.
- When a JavaScript receives an “exit” event (at a waiting point) then it can decide to exit, if it finds it relevant.
- Whenever a JavaScript exits, the event queue and all variables are cleared. The cleanup lasts for one second.
- After exit of a JavaScript the internally queued JavaScripts are requested to be activated (by the Script Handler) in the order originally received.

Example:
Four JavaScripts with the same priority are requested to be scheduled. After that a fifth JavaScript with a higher priority is requested to be executed. The following sequence of events will happen:

- Script #1 with priority 1 is requested to be executed (by the user). Since no JavaScripts are currently executing then the JavaScript is started. The JavaScript waits for an event at some point.
- Script #2 with priority 1 is requested to be executed (by the user). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #3 with priority 1 is requested to be executed (by the user). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #4 with priority 1 is requested to be executed (by the user). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #5 with priority 2 is requested to be executed (by the user). Since a JavaScript with a lower priority is already executing, then the JavaScript is inserted into the internal queue. Also an "exit" event is generated.

- Script #1 receives the "exit" event and exits. Cleanup is performed.
- Script #2 with priority 1 is requested to be executed (by the Script Handler). Since no JavaScripts are currently executing then the JavaScript is started. The JavaScript waits for an event at some point.
- Script #3 with priority 1 is requested to be executed (by the Script Handler). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #4 with priority 1 is requested to be executed (by the Script Handler). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #5 with priority 2 is requested to be executed (by the Script Handler). Since a JavaScript with a lower priority is already executing, then the JavaScript is inserted into the internal queue. Also an "exit" event is generated.

- Script #2 receives the "exit" event and exits. Cleanup is performed.
- Script #3 with priority 1 is requested to be executed (by the Script Handler). Since no JavaScripts are currently executing then the JavaScript is started. The JavaScript waits for an event at some point.
- Script #4 with priority 1 is requested to be executed (by the Script Handler). Since a JavaScript with the same priority is already executing, then the JavaScript is inserted into the internal queue.
- Script #5 with priority 2 is requested to be executed (by the Script Handler). Since a JavaScript with a lower priority is already executing, then the JavaScript is inserted into the internal queue. Also an "exit" event is generated.

- Script #3 receives the "exit" event and exits. Cleanup is performed.
- Script #4 with priority 1 is requested to be executed (by the Script Handler). Since no JavaScripts are currently executing then the JavaScript is started. The JavaScript waits for an event at some point.
- Script #5 with priority 2 is requested to be executed (by the Script Handler). Since a JavaScript with a lower priority is already executing, then the JavaScript is inserted into the internal queue. Also an "exit" event is generated.

- Script #4 receives the "exit" event and exits. Cleanup is performed.
- Script #5 with priority 2 is requested to be executed (by the Script Handler). Since no JavaScripts are currently executing then the JavaScript is started.

10.9.2 Limitations

10.9.2.1 Script sequences

The above-explained scheduling algorithm is very simple and does not easily allow for control of JavaScript sequences. Extreme care must be taken if such sequences are used, in particular in cases where JavaScript activation can interrupt a sequence of JavaScripts.

Example 1:

- Weight is measured. This activates a Weight-1 JavaScript. The Weight-1 JavaScript activates a Weight-2 JavaScript
- BP is measured. This activates a BP-1 JavaScript. The BP-1 JavaScript activates a BP-2 JavaScript.

The JavaScripts have waiting points and responds to "exit" events.

This is ok if the two measurements are performed one after another (BP is not measured until the Weight-2 JavaScript has exited).

Example 2 (same JavaScripts as in example 1):

- Weight is measured. This activates the Weight-1 JavaScript.
- BP is measured. The BP-1 JavaScript is activated (when Weight-1 has finished).
- The Weight-2 JavaScript is activated (when BP-1 has finished).
- The BP-2 JavaScript is activated (when Weight-2 has finished).

This might not be what the user wanted.

10.9.2.2 Events

List of existing events:

- LEvent
- REvent
- SEvent
- IEvent
- PHRASEEvent
- HOSTEvent
- DIALEvent
- CONNECTEvent
- DISCONNECTEvent
- PACKAGEEvent
- TIMEOUTEvent
- EXITEvent

NOTE: It is important to explicitly handle all possible events at script waiting points. If this is not done then it is very easy to get out of synchronization with events. Do not assume (unless you are absolutely sure) without checking on types of events. And be careful when you handle an event that what you do is what you want.

10.10 Initial initialization of RTX337x

This section describes an example start-up initialization of RTX337x from the point where the RTX337x database only has default values for all entries. This could be after an AT+pFS= command.
The description is exemplified with how things could be done, but there are many other possibilities and it is the JavaScript author who determines the exact behavior.

10.10.1 RTX337x Modes

The RTX337x has three modes, FACTORY, PATIENT and NORMAL mode. The current mode is saved in the database. Whenever power is turned on, the RTX337x reads the mode from the database and activates a JavaScript particular for this mode if such a JavaScript is present.
The above lines are the essence of the following description of RTX337x modes and handling of these modes.

The three modes are:

- **FACTORY**
  This is the mode when leaving the factory or after an AT+pFS= command. It is possible to revert to this mode by the key combination “Volume Down, Scroll Down and Right soft button” for 5 sec. The service interface is permanently enabled in this mode and it is possible to use this interface for AT commands. Initialization of the RTX337x can be performed both from the service interface and from the Host (Internet) Server.
  At power on in this mode a FactoryScript is activated if present. The JavaScript has been loaded from the Host Server or loaded via the tty. The name of the JavaScript has been set by means of the AT+pFACTORYSCRIPT command. The FactoryScript is activated either by turning power off and then on again, or by means of the AT+pACTIVATE command.

  In this mode the following JavaScripts must be loaded and their names set:

  FactoryScript

For factory initialization. The name of the JavaScript is set by means of the AT+pFACTORYSCRIPT command (only loaded if not already there – or if it should be replaced by some reason).

PatientScript.
For patient initialization. The name of the JavaScript is set by means of the AT+pPATIENTSCRIPT command.

NormalScript.
For normal use. The name of the JavaScript is set by means of the AT+pNORMALSCRIPT command.

The JavaScripts can be loaded, and their names set, either by means of the tty or by means of the Host Server.
The FactoryScript, when activated, should first determine modem parameters and then initiate server communication by means of a REQUESTREGISTRATION type telegram to the Host Server.
To continue processing properly the Host Server, after having downloaded and set the names of the above mentioned JavaScripts, could download an extension to the factory JavaScript and activate this JavaScript. This should ask the user if a test JavaScript should be downloaded from the Host Server and activated. If this is the case the server is asked to do so. After having performed whatever the test JavaScript wants, the test JavaScript should set the PATIENT mode and ask the user to turn off power.
If no test JavaScript the FactoryScript extension should set the PATIENT mode and ask the user to turn off power.
The PatientScript should present the user with a question stating: "Ready for patient enrolment?".

- **PATIENT**
  When the RTX337x is powered up in this mode, the PatientScript is automatically activated. The service interface is disabled (in External device mode).
  When the user answers OK to "Ready for patient enrolment?", the JavaScript should determine modem parameters (partly automatic by asking the modem and partly by user interaction via the JavaScript). These parameters are saved in the database.
  After this the PatientScript should request patient configuration from the Host Server. After having received this, the PatientScript should set the NORMAL mode and exit.
  The NormalScript is now activated automatically when the PatientScript exits.
  The NormalScript should present the user with the default screen showing the clock.

- **NORMAL**
  When the RTX337x is powered up in this mode, the NormalScript is automatically activated. The service interface is disabled (in external device mode).
  This is the normal mode for the RTX337x. Full functionality of the RTX337x is active in this mode.

10.10.2 RTX337x Power On

Whenever power is turned on the RTX337x tries to send a telegram to the Host Server with the telegram type HEREIAM. The value field of this telegram identifies the current mode of the RTX337x. In FACTORY mode it is FACTORY, in PATIENT mode it is PATIENT and in NORMAL mode it is NORMAL. This gives the Host Server the possibility to communicate with the RTX337x, particularly for configuration purposes.

10.10.3 Example Start-up Initialization of RTX337x

This section will give a step by step description of an example start-up initialization of the RTX337x.

10.10.3.1 JavaScripts

The following JavaScripts will be needed:

- Factory
  This is the factory JavaScript. The JavaScript performs the following functions:
  - Determine modem parameters in cooperation with the user.
  - Register on Host Server. Sends a telegram with the type REQUESTREGISTRATION.
  - Asks user if a factory test should be performed.
  - If a factory test should be performed then request the factory test JavaScript from the Host Server i.e. send a telegram with the telegram type REQUESTFACTORYTEST to the Host Server. The JavaScript then awaits activation of the factory test from the Host Server (AT+pACTIVATE).
  - If a factory test should not be performed then the RTX337x mode is set to PATIENT mode and the user is told to turn off power to the RTX337x.
- FactoryT
  This is the factory test JavaScript. The JavaScript performs the following functions.
  - Test buttons etc. in cooperation with user.
  - Set the RTX337x mode to PATIENT mode and tell user to turn off power to the RTX337x.
- Patient
  This is the patient JavaScript. The JavaScript performs the following functions:
  - Asks the user if he is ready for patient enrolment.
  - Determines modem parameters in cooperation with the user.
  - Request patient initialization from the Host Server i.e. send a telegram with the type REQUESTPATIENTINITIALIZATION to the Host Server.
  - Await patient initialization configuration of the RTX337x.
  - Host Server uploads modem parameters.
  - Set RTX337x mode to NORMAL and exit.
- Normal
  This is the default JavaScript. The JavaScript shows the current time.

10.10.3.2 Step by step description

The RTX337x database is assumed to begin with default values.

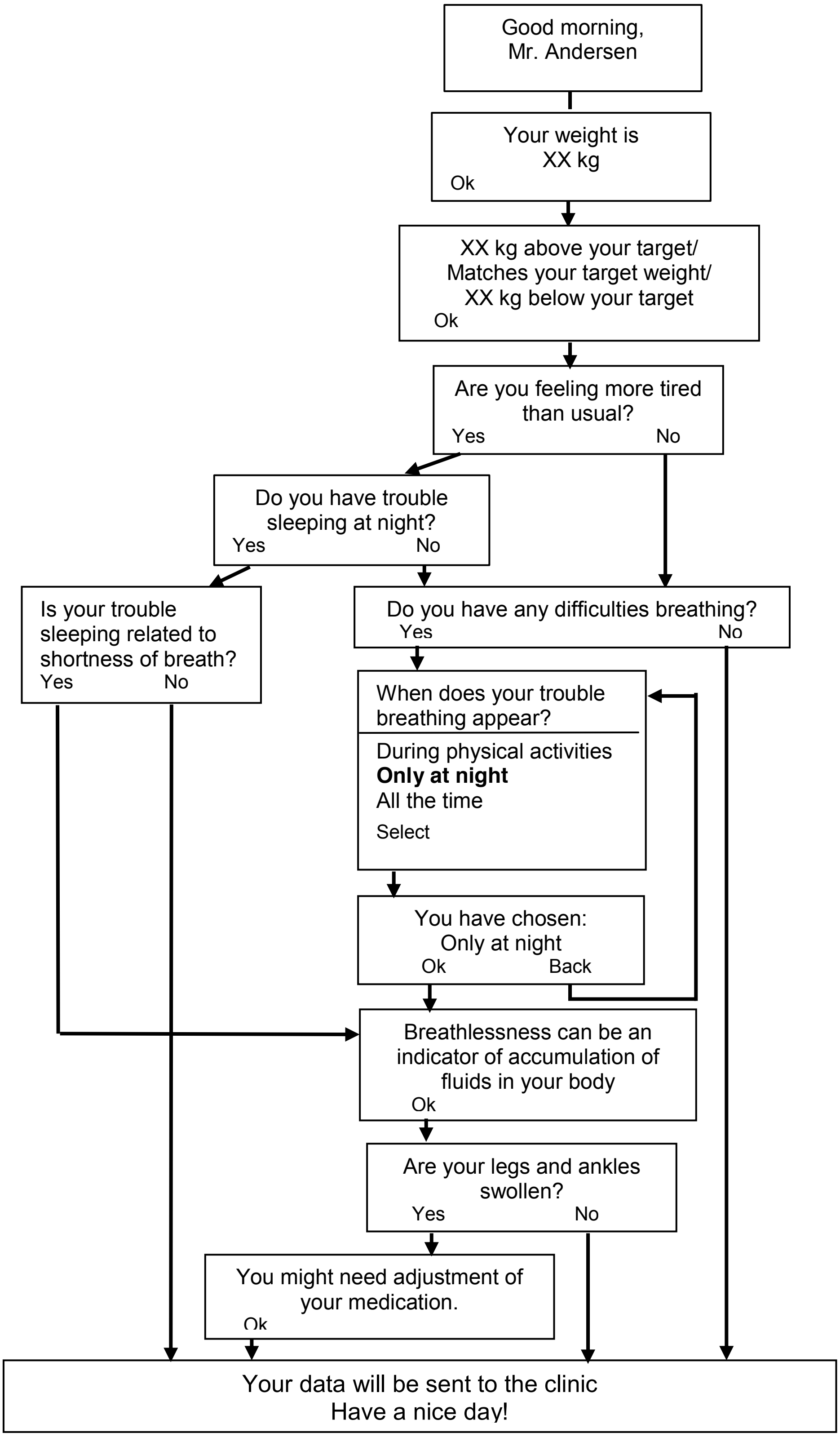
1. Assure telephone line is connected.
2. Turn power on for the RTX337x.
3. The RTX337x sends a HEREIAM telegram to the Host Server with the value field FACTORY.
4. The Host Server does initial configuration of the RTX337x. This includes downloading the Factory JavaScript, the Patient JavaScript and the Normal JavaScript. And all voice files used by these JavaScripts. It also sets these JavaScripts (AT+pFACTORYSCRIPT=Factory, AT+pPATIENTSCRIPT=Patient,

AT+pNORMALSCRIPT=Normal). The RTX337x mode is set to FACTORY (AT+pMODE=FACTORY).

5. The Host Server activates the Factory script (AT+pACTIVATE=Factory,….). Alternatively power can be turned off and then on again.
6. The Factory script is now active. It performs as described in section 10.10.1. It ends with setting the RTX337x mode to PATIENT and a request to the user to turn power off.
7. Turn power off.
8. Turn power on.
9. The Patient script is now active. It performs as described in section 10.10.1. It ends with setting the RTX337x mode to NORMAL and then exits. This activates the Normal JavaScript.
10. The Normal script is now active. The RTX337x is running in NORMAL mode.

10.11 JavaScript Examples

The first example is a JavaScript activated when measurements are received from a weight scale. The Question Tree structure is illustrated below:


Good morning, Mr. Andersen
Your weight is XX kg
Ok
XX kg above your target/ Matches your target weight/ XX kg below your target
Ok
Are you feeling more tired than usual?
Yes
No
Do you have trouble sleeping at night?
Yes
No
Is your trouble sleeping related to shortness of breath?
Yes
No
Do you have any difficulties breathing?
Yes
No
When does your trouble breathing appear?
During physical activities
Only at night
All the time
Select
You have chosen: Only at night
Ok
Back
Breathlessness can be an indicator of accumulation of fluids in your body
Ok
Are your legs and ankles swollen?
Yes
No
You might need adjustment of your medication.
Ok
Your data will be sent to the clinic
Have a nice day!

The second example is a JavaScript displaying the time and a 'logo' (used as Normal script).
The examples are meant to illustrate main features; they might not match the current version available in the Demonstration scripts on the eSupport site.

10.11.1 JavaScript example for Weight Scale

```
//The script implements handling of weight received from a weight scale.
//
//
//
//#include VoiceNumberFunction
/**
 * This function constructs the number in numberText from files containing
 * the components, ie 145 plays the soundfiles 100, 40, 5.
 * @param {String} numberText The number which should be played
 * @param {boolean} LastSound Indicates whether the number is at the end of
a
 * spoken phrase or not, default is not
 *
 * @requires GetNumberName
 *
 * @throws Error Detailed Error message.
 */

function VoiceNumber(numberText, last_sound)
{
    var hundrets = 0;
    var numberValue = 0;
    var tens = 0;
    var ones = 0;
    var j = 0;
    var i = 0;
    var LastSound = 0;
    var nmb = new Array();
    var RmAnd = 0;
    var res;

    if (last_sound)
          LastSound = 1;

    numberValue = parseInt(numberText,10);

//    gw.Log("Debug numbervalue: " + numberValue);

    hundrets = Math.floor(numberValue / 100);
    numberValue %= 100;
    tens = Math.floor(numberValue / 10);
    numberValue %= 10;
    ones = numberValue;
    j = 0;

    if (hundrets != 0) {
        nmb[j] = GetNumberName("" + (hundrets*100));
        j++;
        nmb[j] = "And";
        RmAnd = ++j;
    }
    if (tens >= 2) {  // 20 - 30 - 40 - etc.
        nmb[j] = GetNumberName("" + (tens*10));
        j++;
    }

    if (ones != 0 && tens != 1) { // 1-9 + 21-29 + 31-39 etc.
        nmb[j] = GetNumberName("" + ones);
        j++;
    } else if (ones != 0 && tens == 1) { // 11-19
        nmb[j] = GetNumberName("" + ((tens*10)+ones));
        j++;
    } else if (ones == 0 && tens == 1) { // 10
        nmb[j] = GetNumberName("" + (tens*10));
        j++;
    } else if (ones == 0 && tens == 0 && hundrets == 0) { // 0
        nmb[j] = GetNumberName("0");
        j++;
    }
//    gw.Log("Debug numbervalue: " + nmb[0] + " " + nmb[1] + " " + nmb[2]);

    if (RmAnd == j) {
        j--;
    }
    for (i = 0; i < j; i++){
        if (i == j-1)
              res = gw.PlaySound(LastSound, nmb[i]);
        else
              res = gw.PlaySound(0, nmb[i]);
        if (res < 0)
            throw new Error("VoiceNumberFunction - gw.PlaySound failed
trying to play: " + nmb[i]);
    }
}

/**
 * This function returns the ASCII id for a number. NOTE! This currently
 * supports:
 * <ul><li>0-19</li>
 * <li>20-90 (steps of 10) </li>
 * <li>100-300 (steps of 100). </li>
 * <li>. (decimal point)</li></ul>
 *
 * @param {String} num The number to return the name of.
 *
 * @return {String} ASCII name of the phrasefile containing this number, or
 * an empty string in case of no match.
 */
function GetNumberName(num)
{
    var nmbId = "";

    switch (num)
    {
        case "0":
            nmbId = "Zero";
            break;

        case "1":
            nmbId = "One";
            break;

        case "2":
            nmbId = "Two";
            break;

        case "3":
            nmbId = "Three";
            break;

        case "4":
            nmbId = "Four";
            break;

        case "5":
            nmbId = "Five";
            break;

        case "6":
            nmbId = "Six";
            break;

        case "7":
            nmbId = "Seven";
            break;

        case "8":
            nmbId = "Eight";
            break;

        case "9":
            nmbId = "Nine";
            break;

        case "10":
            nmbId = "Ten";
            break;

        case "11":
            nmbId = "Eleven";
            break;

        case "12":
            nmbId = "Twelve";
            break;

        case "13":
            nmbId = "Thirteen";
            break;

        case "14":
            nmbId = "Fourteen";
            break;

        case "15":
            nmbId = "Fifteen";
            break;
```

```
case "16":
    nmbId = "Sixteen";
    break;

case "17":
    nmbId = "Seventeen";
    break;

case "18":
    nmbId = "Eighteen";
    break;

case "19":
    nmbId = "Nineteen";
    break;

case "20":
    nmbId = "Twenty";
    break;

case "30":
    nmbId = "Thirty";
    break;

case "40":
    nmbId = "Fourty";
    break;

case "50":
    nmbId = "Fifty";
    break;

case "60":
    nmbId = "Sixty";
    break;

case "70":
    nmbId = "Seventy";
    break;

case "80":
    nmbId = "Eighty";
    break;

case "90":
    nmbId = "Ninety";
    break;

case "100":
    nmbId = "Hundred";
    break;

case "200":
    nmbId = "TwoHundred";
    break;

case "300":
    nmbId = "3Hundred";
    break;
```

```
        case ".":
            nmbId = "Point";
            break;

        default:
            nmbId = "";
            break;
    }

    return (nmbId);
}

//#include QuestionFunction
/**
 * This function shows the text in txtName and awaits that the
 * user presses the yes or the no button. <br />
 * NOTE! This function use timeouttype 0 if the timeout argument is
 * present.
 *
 * @param {String} txtName The name of the phrasefile to display.
 * @param {int} timeout Timeout for user input (default is no timeout).
 * @param {boolean} sound Whether audio should be played or not (default is
 * true)
 * @param {int} cnf How the text should be shown (default is 0).
 * @param {Array} button Array containing the phrase files to be used as
 * buttons, must have 2 entries.
 *
 * @returns {gw.*Event} LEFTEvent (Yes), RIGHTEvent (No) or if a timeout is
 * specified TIMEOUTEvent (timeout).
 *
 * @requires WaitForEvent
 *
 * @throws Error Detailed Error message.
 */

function Question(txtName, timeout, sound, cnf, button)
{
    //Initialize.
    var type, res;
    var events = new Array();
    var activeButtons = 0;
    var pressedButton = 0;
    var resp = new Array();

    if (sound === undefined)
        sound = true;

    if (timeout === undefined)
        timeout = 0;
    if (cnf === undefined)
          cnf = 0;
    if (!(button instanceof Array))
    {
        button = new Array();
        //Soft buttons (yes and no) and await user response.
        button[0] = "Yes";
        button[1] = "No";
    }
    //determine active buttons
    if (button[0].length > 0)
    {
        activeButtons += 1;
        events = events.concat(gw.LEvent());
    }
    if (button[1].length > 0)
    {
        activeButtons += 2;
        events = events.concat(gw.REvent());
    }
    //Question.
    gw.AddToResp(gw.GetText(txtName) + " \n");  //User response.
    res = gw.ShowText(cnf, txtName);        //Show question.
    if (res < 0)
        throw new Error("Question - gw.ShowText failed trying to show: " + 
txtName);
    if (sound)
    {
        res = gw.PlaySound(1, txtName);       //Voice question.
        if (res < 0)
              throw new Error("Question - gw.PlaySound failed trying to 
play: " + txtName);
        WaitForEvent(resp, gw.PHRASEEvent());          //Wait for speak to 
finish.
    }

    res = gw.ShowButtons(0, button);
    if (res < 0)
          throw new Error("Question - gw.ShowButtons failed trying to 
display buttons: " + button);

    if (timeout > 0)
    {
        events = events.concat(gw.TIMEOUTEvent());
        gw.SetTimeout(0, timeout);
    }
    do {
        gw.ActivateButtons(activeButtons);                  //Activate 
buttons.
        type = WaitForEvent(resp, events);      //Wait for button press.
        if (type == gw.TIMEOUTEvent() && timeout > 0 && resp[0] == 0)
              return (type); //return the timeout
    } while (type == gw.TIMEOUTEvent()); //throw away other timeout events.
    gw.ClrTimeout(0);
    switch (type)
    {
        case gw.LEvent():
          pressedButton = 0;
          break;

        case gw.REvent():
          pressedButton = 1;
          break;

        default:
          throw new Error("Question - Received unexpected eventtype");
          break;
    }
    //Acknowledge to user.
    gw.AddToResp(gw.GetText(button[pressedButton]) + " \n");      //User
response.
    button[1 - pressedButton] = "";
    res = gw.ShowButtons(0, button);
    if (res < 0)
          throw new Error("Question - gw.ShowButtons failed trying to
display buttons: " + button);
    if (sound)
    {
        res = gw.PlaySound(1, button[pressedButton]);
        if (res < 0)
              throw new Error("Question - gw.PlaySound failed trying to
play: " + button[i]);
        WaitForEvent(resp, gw.PHRASEEvent());
    }

    //Return activated button.
    return (type);
}

//#include ShowOKFunction
/**
 * Display an optional phrase file and show OK button and wait for response
from user,
 * then play the sound for OK.
 *
 * @param {String} txtName Name of the phrase to display/play. This
 * parameter is optional, if it is left out only the OK button is showed.
 * @param {int} timeout Optional timeout, if defined and not 0 then the
 * question is only shown for the timeout duration and if no button press
is
 * registred the function returns a timeout event.
 * @param {boolean} sound Whether audio should be played or not (default is
 * true)
 * @param {int} cnf How the text should be shown (default is 0).
 *
 * @requires WaitForEvent
 *
 * @throws Error Detailed Error message.
 */
function ShowOK(txtName, timeout, sound, cnf) {

    var resp = new Array();
    var res;

    if (timeout === undefined)
        timeout = 0;

    if (sound === undefined)
        sound = true;

    if (txtName) //if txtName is supplied show it.
    {
        //Question.
        gw.AddToResp(gw.GetText(txtName) + " \n");    //User response.
        res = gw.ShowText(0, txtName);         //Show question.
        if (res < 0)
```

```
            throw new Error("ShowOK - gw.ShowText failed trying to display:
" + txtName);
        if (sound)
        {
            res = gw.PlaySound(1, txtName);       //Voice question.
            if (res < 0)
                  throw new Error("ShowOK - gw.PlaySound failed trying to
play: " + txtName);

            WaitForEvent(resp, gw.PHRASEEvent());          //Wait for speak
to finish.
        }
    }

    var button = new Array();
    button[0] = "Ok";
    button[1] = "";
    res = gw.ShowButtons(0, button);
    if (res < 0)
          throw new Error("ShowOK - gw.ShowButtons failed trying to display
buttons: " + button);

    gw.ActivateButtons(1);

    WaitForEvent(resp, gw.LEvent(), timeout);

    //Voice Ok and wait for voice to finish.
    if (sound)
    {
        res = gw.PlaySound(1, button[0]);
        if (res < 0)
              throw new Error("ShowOK - gw.PlaySound failed trying to play
OK");
        WaitForEvent(resp, gw.PHRASEEvent());
    }
    return;
}

//#include QuestionScrollFunction
/**
 * Shows a Question with a series of possible answers shown in a menu
below.
 * @param {String} txtName The name of the phrasefile with the question
 * @param {Array} menu Array of names of phrasefiles containing the
possible
 * @param {int} cnf How the menu should be shown, values are additive and
 * the part in capital letters are references to Constants.js which can be
 * used instead of the value: <ul><li>Size</li>
 *      <ul><li>6 (MENUHEAD) = Single line question</li>
 *      <li>7 (LARGEMENUHEAD) = Two line question (default)</li></ul>
 * <li>Prefix</li>
 *      <ul><li>0 (NOPREFIX) = no prefix (default)</li>
 *      <li> 256 (PREFIX) = prefix</li></ul>
 * </ul>
 * @param {int} select The index menu which should be displayed as the
initial
 * choice.
 * @param {boolean} confirm Indicates whether the user should confirm the
 * choice or not (default).
 * @param {int} timeout Timeout for user input (default is no timeout).
 * @param {boolean} sound Whether audio should be played or not (default is
 * true)
 *
 * @return {int} The index of the selected answer.
 *
 * @requires Menu
 *
 * @throws Error Detailed Error message.
 */

function QuestionScroll(txtName, menu, cnf, select, confirm, timeout,
sound)
{
    var type;
    var act;
    var button = new Array();
    button[0] = "Select";
    button[1] = "";


    if (!cnf)
          cnf = 7; //default to cnf = 7, Large menu header.
    if (!select)
          select = 0;     // select entry 0 as default  txtName = "When";
          act = 13;       //All buttons but right (No)active.

    do{
        value = Menu(txtName, menu, cnf, select, button, act, false,
timeout, sound);

        gw.ClrScr();
        if (confirm)
              type=QuestionBack(menu[value], timeout, sound);
//confirmation
        else
              break; //if no confirmation is break
    }
    while(type==gw.REvent());


    if (value >=0 && value < menu.length)
          gw.AddToResp("Chosen parameter: " + menu[value] + " \n");
      //User response.
    else
          gw.AddToResp("Error in selection!!!\n");

    return value;
}


/**
 * This function shows a phrasefile and awaits that the user presses
 * the Ok or the Undo button, NOTE this function is internal for
QuestionScroll.
 * @param {String} txtName The name of the phrasefile.
 *
 * @returns {gw.*Event} LEFTEvent (Yes), RIGHTEvent (No) or if a timeout is
 * specified TIMEOUTEvent (timeout).
 *
 * @requires WaitForEvent
 *
 * @requires Text
 */

function QuestionBack(txtName, timeout, sound)
{
    //Initialize.
    var resp = new Array();
    var res;

    if (timeout === undefined)
        timeout = 0;

    if (sound === undefined)
        sound = true;

    //Question.
    Text(3, "YourChoice", true);           //Confirmation of choice
    gw.AddToResp(gw.GetText(txtName) + " \n");  //User response.
    res = gw.ShowText(4, txtName);         //Show question.
    if (res < 0)
        throw new Error("QuestionBack - gw.ShowText failed trying to show:
" + txtName);
    if (sound)
    {
        res = gw.PlaySound(1, txtName);        //Voice question.
        if (res < 0)
              throw new Error("QuestionBack - gw.PlaySound failed trying to
play: " + txtName);
        WaitForEvent(resp, gw.PHRASEEvent());            //Wait for speak to
finish.
    }

    //Soft buttons (yes and no) and await user response.
    var button = new Array();
    button[0] = "Ok";
    button[1] = "Undo";
    res = gw.ShowButtons(0, button);
    if (res < 0)
          throw new Error("QuestionBack - gw.ShowButtons failed trying to
display buttons: " + button);
    if (timeout > 0)
        gw.SetTimeout(0, timeout);
    do {
        gw.ActivateButtons(3);                   //Activate buttons.
        var type = WaitForEvent(resp);     //Wait for button press.
        if (type == gw.TIMEOUTEvent() && timeout > 0 && resp[0] == 0)
              return (type); //return the timeout
    } while (!(type == gw.LEvent() || type == gw.REvent()));
    gw.ClrTimeout(0);
    //Acknowledge to user.
    var i = (type == gw.LEvent()) ? (0) : (1);
    gw.AddToResp(gw.GetText(button[i]) + " \n");      //User response.
    button[1 - i] = "";
    res = gw.ShowButtons(0, button);
    if (res < 0)
```

```
          throw new Error("QuestionBack - gw.ShowButtons failed trying to 
display buttons: " + button);
    if (sound)
    {
        res = gw.PlaySound(1, button[i]);
        if (res < 0)
              throw new Error("QuestionBack - gw.PlaySound failed trying to 
play: " + button[i]);
        WaitForEvent(resp, gw.PHRASEEvent());
    }

    //Return activated button.
    return (type);
}

//#include WaitForEventFunction
/**
 * Waits for the specified event or if no event is specified for any event.
 * If asked to wait for an EXITEvent and one has been registred earlier it
 * returnes the event immediately.
 * @param {Array} resp An array which will contain additional information 
about the event.
 * @param {gw.*event} event The event to wait for, if no event is specified 
the
 * first event is returned (this might include a remembered EXITEvent). It
 * is allowed to pass an Array of events in which case the function will
 * wait for one on of the events in the array.
 *
 * @return {gw.*event} The first event the user asked to wait for.
 *
 * @requires exitEventDetected defined as variable at global scope.
 *
 * @throws Error Detailed Error message.
 *
 */
function WaitForEvent(resp, event)
{
    var eventType;
    //if resp is not an array ensure it is one
    if (resp instanceof Array != true)
        throw new Error("WaitForEvent - first parameter must be an Array");

    if (event === undefined)
    {
        if (exitEventDetected == 1) {
            exitEventDetected = 0;
            return gw.EXITEvent();
        } else {
            eventType = gw.WaitForEvent(resp);
            if (eventType < 0)
            {
                throw new Error("WaitForEvent - gw.WaitForEvent returned -
1");
            }
            return eventType;
        }
    }
    else
    {
        do {
            if (IsEventIn(event, gw.EXITEvent()) && exitEventDetected == 1)
            {
                exitEventDetected = 0;
                return gw.EXITEvent();
            }
            else
            {
                eventType = gw.WaitForEvent(resp);
                if (IsEventIn(event, eventType))
                {
                    return eventType;
                }
                else
                {
                    if (eventType == gw.EXITEvent())
                    {
                        exitEventDetected = 1;
                    }
                    else if (eventType < 0)
                    {
                        throw new Error("WaitForEvent - gw.WaitForEvent
returned -1");
                    }
                }
            }
        } while (1);
    }
}

/**
 * Helper function for WaitForEvent()
 *
 */
function IsEventIn(event, eventType)
{
    if (event instanceof Array)
    {
        if (event.length == 0)
            throw new Error("WaitForEvent - called with empty array");
        var i;
        for (i = 0; i < event.length; i++)
        {
            if (event[i] === eventType)
                return true;
        }
        return false; //no match
    }
    else
        return (event === eventType);
}

//#include MenuFunction
/**
 * This function shows a menu and awaits the user to select
 * one of the entries. Based on this it generates the proper event.
 *
 * @param {String} txtName Contains the name of the phrasefile or the
string with the
```

```
 * question.
 * @param {Array} menu Contains the names of the phrasefiles to be used or
 * the literal strings.
 * @param {int} cnf How the menu should be shown, values are additive and
 * the part in capital letters are references to Constants.js which can be
 * used instead of the value: <ul><li>Size</li>
 *      <ul><li>6 (MENUHEAD) = Single line question</li>
 *      <li>7 (LARGEMENUHEAD) = Two line question (default)</li></ul>
 * <li>Prefix</li>
 *      <ul><li>0 (NOPREFIX) = no prefix (default)</li>
 *      <li> 256 (PREFIX) = prefix</li></ul>
 * </ul>
 * @param {int} select The index in menu which should be the initially
 * chosen menu item.
 * @param {Array} button Array with the button text which should be shown,
 * either as reference or as literals.
 * @param {int} act Which buttons should be active
 * @param {boolean} LitMenu Determines whether the content of txtName, menu
 * and button are phrasefile names or literal strings. Default is
phrasefile names.
 * @param {int} timeout Timeout for user input (default is no timeout).
 * @param {boolean} sound Whether audio should be played or not (default is
 * true)
 *
 * @returns -1 if the user presses the Right button otherwise the selected
menu entry.
 *
 * @requires WaitForEvent
 *
 * @throws Error Detailed Error message.
 */

function Menu(txtName, menu, cnf, select, button, act, LitMenu, timeout,
sound)
{
    var resp, type, res;
    var tmp;
    //Initialize.
    resp = new Array();
    gw.ClrScr();

    if (timeout === undefined)
        timeout = 0;

    if (sound === undefined)
        sound = true;

    //ensure cnf has default value for menu
    tmp = cnf & 0xF;
    if (tmp != 6 && tmp != 7)
    {
        cnf = (cnf & (~0xF)); //remove bits 0-4
          cnf += 7; //add default
    }

    //Show headline text.
    if (!LitMenu)
        res = gw.ShowText(cnf, txtName);
    else
```

```
        res = gw.ShowTextL(cnf, txtName);

    if (res < 0)
          throw new Error("Menu - gw.ShowText failed trying to show: " +
txtName);

    if (!LitMenu && sound)
    {
        res = gw.PlaySound(1, txtName);
        if (res < 0)
              throw new Error("Menu - gw.PlaySound failed trying to play: "
+ txtName);
        type = WaitForEvent(resp, gw.PHRASEEvent());
    }
    //determine menu type
    if ((cnf & 0xF) == 7)
          cnf |= 1<<16; //ensure the menu is drawn to match chosen header

    //ensure Menu text is written left alligned
    cnf |= 1<<20;

    //Show menu.
    if (!LitMenu)
          res = gw.ShowMenu(cnf, select, menu);
    else
          res = gw.ShowMenuL(cnf, select, menu);

    if (res < 0)
          throw new Error("Menu - gw.ShowMenu failed trying to show menu: "
+ menu);

    //Show soft buttons.
    if (!LitMenu)
          res = gw.ShowButtons(0, button);
    else
          res = gw.ShowButtonsL(0, button);

    if (res < 0)
          throw new Error("Menu - gw.ShowButtons failed trying to show: " +
button);

    //Play initial selection sound.
    if (!LitMenu && sound)
    {
        res = gw.PlaySound(1, menu[select]);
        if (res < 0)
              throw new Error("Menu - gw.PlaySound failed trying to play: "
+ menu[select]);
        type = WaitForEvent(resp, gw.PHRASEEvent());
    }

    //Loop until the user makes a selection.
    do
    {
        //Activate buttons.
        gw.ActivateButtons(act);

        //Wait for an event.
        type = WaitForEvent(resp);
        if (type == gw.SEvent()) {
            if (!LitMenu && sound)
            {
                res = gw.PlaySound(1, menu[resp[0]]);
                if (res < 0)
                      throw new Error("Menu - gw.PlaySound failed trying to
play: " + menu[resp[0]]);
                do {
                    type = gw.WaitForEvent(resp);
                } while (type != gw.PHRASEEvent());
            }
        }
    } while ((type != gw.LEvent()) && (type != gw.REvent()));

    //Return proper event.
    if (type == gw.REvent())
          resp[0] = -1;

    return (resp[0]);
}

//#include TextFunction
/**
 * This function shows the content of a phrase filename and plays the
audio.
 * @param {int} cnf How the text should be shown on the screen.
 * @param {String} txt The name of the phrasefile to be used, or the string
 * to be shown.
 * @param {boolean} last_sound whether the string is in the end of a spoken
 * sentence
 * @param {boolean} sound Whether audio should be played or not (default is
 * true)
 * @param {boolean} literal Whether the text to be displayed is a
phrasefile
 * or a text string (default is phrase file).
 *
 * @requires WaitForEvent
 *
 * @throws Error Detailed Error message.
 */
function Text(cnf, txt, last_sound, sound, literal)
{
    //Initialize.
    var resp = new Array();
    var res;
    if (literal === undefined)
        literal = false;

    if (sound === undefined)
        sound = true;

    if (literal)
        sound = false; //no sound on literals

    if (!literal)
          res = gw.ShowText(cnf, txt);  //Show message.
    else
          res = gw.ShowTextL(cnf, txt);
    if (res < 0)
```

```
        throw new Error("Text - gw.ShowText failed trying to show: " +
txtName);
    if (sound)
    {
        if(last_sound)
        {
            res = gw.PlaySound(1, txt);   //Voice message.
            if (res < 0)
                  throw new Error("Text - gw.PlaySound failed trying to
play: " + txtName);
            WaitForEvent(resp, gw.PHRASEEvent());      //Wait for speak to
finish.
        }
        else
        {
            res = gw.PlaySound(0, txt);   //Voice message.
            if (res < 0)
                  throw new Error("Text - gw.PlaySound failed trying to
play: " + txtName);
        }
    }
}

//
//The parameters for this script is:
//    param[0] :   The current weight.
//    param[1] :   The measurement time (seconds since 1-jan-1970).
//    param[2] :   The measurement type (0 <=> lbs; 1 <=> kgs).
//    param[3] :   BatteryStatus  (0-255 (See Com serires - version 2.3.pdf
//    for details)).
//    param[4] :   SaveStatus (1=success, 0=No save (buffer full)).
//    param[5] :   Upper weight limit.
//    param[6] :   Lower weight limit.

function main()
{
    //Initialize.
    resp = new Array();
    gw.ClrScr();

    //Get parameters.
    var weight = gw.GetScriptPar(0);
    var mesTime = gw.GetScriptPar(1);
    var mesType = gw.GetScriptPar(2);
    var weightU = gw.GetScriptPar(5);
    var weightL = gw.GetScriptPar(6);

    var weight = Math.round(weight);

    //Initial salute ----------------------.
    var time = gw.GetTime();
    var d = new Date(time * 1000);
    var clock = d.getUTCHours();

    //Get proper salute.
    var salute = (clock < 12) ? ("GoodM") : ((clock < 18) ?
            ("GoodA") : ("GoodE"));

    var patientName = "PatNam1";    //Mr.Andersen
    //Display salute.
    var text = gw.GetText(salute) + ","+"                     " +
gw.GetText(patientName);
    gw.ShowTextL(0, text);

    //User response: Patient name and script parameters.
    gw.AddToResp(gw.GetText(patientName) + " \n");
    gw.AddToResp("W: " + weight + " Time: " + mesTime + " Type: " +
            mesType + " U: " + weightU + " L: " + weightL + " \n");

    //Voice salute and wait for speak to finish.
    gw.PlaySound(0, salute);
    gw.PlaySound(1, patientName);
    WaitForEvent(resp, gw.PHRASEEvent());

    gw.SetTimeout(1,3);
    WaitForEvent(resp, gw.TIMEOUTEvent());

    //Weight ----------------------------.
    //Display weight.
    gw.ClrScr();
    var weightToDay = "WToDay";
    var units = (mesType == 0) ? ("Pounds") : ("Kilograms");
    var unit = (mesType == 0) ? ("Pound") : ("Kilogram"); //Pound currently
                                                       //unavailable
    weightText = weight.toFixed();
    if (Math.abs(weightText) == 1)
        var weightUnit = unit;
    else
        var weightUnit = units;

    text =gw.GetText(weightToDay) + " " + weightText + " " +
            gw.GetText(weightUnit);
    gw.ShowTextL(0, text);

    //Voice weight.
    gw.PlaySound(0, weightToDay);
    VoiceNumber(weightText,0);
    gw.PlaySound(1, weightUnit);
    WaitForEvent(resp, gw.PHRASEEvent());

    ShowOK();

    //Weight deviation ---------------------.
    //Display weight deviation.

    gw.ClrScr();
    var weightD = (weight > weightU) ? (weight - weightU) :
            ((weight < weightL) ? (weightL - weight) : (0));
    var weightDText = weightD.toFixed();
    if (Math.abs(weightDText) == 1)
        var weightDUnit = unit;
    else
        var weightDUnit = units;

    //If then else, if then else
    var target = (weight > weightU ) ? ("ATarget") :
            ((weight < weightL) ? ("BTarget") : ("MTarget"));
    if ((weight > weightU) || (weight < weightL))
          text = weightDText + " " + gw.GetText(weightDUnit) + " " +
                  gw.GetText(target);
    else
          text = gw.GetText(target);

    gw.ShowTextL(0, text);

    //Voice weight deviation.
    if ((weight > weightU) || (weight < weightL))
    {
        VoiceNumber(weightDText,0);
        gw.PlaySound(0, weightDUnit);
    }

    gw.PlaySound(1, target);
    WaitForEvent(resp, gw.PHRASEEvent());

    ShowOK();


    QuesWS();
}

function QuesWS()
{

    //Question:
    gw.ClrScr();
    Tired = "Tired";
    type = Question(Tired);

    //If 'Yes' button is pressed as response to Tired question
    if (type==gw.LEvent())
    {
        gw.ClrScr();
        Sleeping = "Sleeping";
        type = Question(Sleeping);

        if (type==gw.LEvent())
        {
            gw.ClrScr();
            TroubleSP = "TroubleSP";
            type = Question(TroubleSP);

            if (type==gw.LEvent())
            {
                gw.ClrScr();
                Indicator = "Indicator";
                type = ShowOK(Indicator);

                gw.ClrScr();
                Swollen = "Swollen";
                type = Question(Swollen);

                if (type==gw.LEvent())
                {
                    gw.ClrScr();
                    Adjust = "Adjust";
                    type = ShowOK(Adjust);
                }
            }
        }
        else{
            gw.ClrScr();
            Breathing = "Breathing";
            type = Question(Breathing);


            if (type==gw.LEvent())
            {
                gw.ClrScr();
                // Questionmenu: When do you have trouble breathing?.
                txtName = "When";
                menu = new Array();                       //Scroll options
                menu[0] = "PhysActiv";
                menu[1] = "Night";
                menu[2] = "AllTime";

                QuestionScroll(txtName, menu, 7, 0, true);


                gw.ClrScr();
                Indicator = "Indicator";
                type = ShowOK(Indicator);

                gw.ClrScr();
                Swollen = "Swollen";
                type = Question(Swollen);

                if (type==gw.LEvent())
                {
                    gw.ClrScr();
                    Adjust = "Adjust";
                    type = ShowOK(Adjust);
                }
            }
        }
    }


    //if 'No' button is pressed as response to Tired question
    else
    {
        gw.ClrScr();
        Breathing = "Breathing";
        type = Question(Breathing);

        if (type==gw.LEvent())
        {
            gw.ClrScr();
            // Questionmenu: When do you have trouble breathing?.
            txtName = "When";
            menu = new Array();                     //Scroll options
            menu[0] = "PhysActiv";
            menu[1] = "Night";
            menu[2] = "AllTime";;
```

```
            QuestionScroll(txtName, menu, 7, 0, true);

            gw.ClrScr();
            Indicator = "Indicator";
            type = ShowOK(Indicator);

            gw.ClrScr();
            Swollen = "Swollen";
            type = Question(Swollen);

            if (type==gw.LEvent())
            {
                gw.ClrScr();
                Adjust = "Adjust";
                type = ShowOK(Adjust);

          }
        }
    }


    //StatusMessage: Your will be sent to the clinic--------.
    gw.ClrScr();
    DataSentClinic = "DataClinic";
    Niceday = "NiceDay";

    gw.ShowText(1, DataSentClinic);
    gw.PlaySound(1,DataSentClinic);    //Voice question.
    gw.WaitForEvent(resp);

    gw.SetTimeout(1,1);
    gw.WaitForEvent(resp);

    gw.ShowText(2, Niceday);
    gw.PlaySound(1,Niceday);           //Voice question (last sound).
    gw.WaitForEvent(resp);             //Wait for speak to finish.
}


//Execute script.
exitEventDetected = 0;
gw.ClrResp();
main();
gw.SendRespToServer();
```

10.11.2 JavaScript example for Time and Logo

```
//The script implements handling of the default normal mode for the
RTX337x.
//
//The script shows the current time on the display and awaits that someone
//wants to execute another script.
//
//#include Constants
//Text windows
SINGLE        = 0;
DUALUP        = 1;
DUALLOW       = 2;
TRIPLEUP      = 3;
```

```
TRIPLEMID      = 4;
TRIPLELOW      = 5;
MENUHEAD       = 6;
LARGEMENUHEAD = 7;

//different fonts
NORMALFNT      = (0 << 4);
BIGFNT         = (1 << 4);
SMALLFNT       = (2 << 4);
SMALLSLIMFNT  = (3 << 4);

//prefix in menu/infos
NOPREFIX       = (0 << 8);
PREFIX         = (1 << 8);

//blinking text
NOBLINK        = (0 << 12);
BLINK          = (1 << 12);

//menu/info layout
MENUBODY       = (0 << 16);
LARGEMENUBODY = (1 << 16);
INFOBODY       = (2 << 16);
LARGEINFOBODY = (3 << 16);

//Alignment
ALIGNCENTER   = (0 << 20);
ALIGNLEFT      = (1 << 20);
ALIGNRIGHT     = (2 << 20);

//buttons
LEFTBUTTON     = (1 << 0)
RIGHTBUTTON   = (1 << 1)
UPBUTTON       = (1 << 2)
DOWNBUTTON     = (1 << 3)
INFOBUTTON     = (1 << 4)


function main()
{
      //Initialize.
      var resp = new Array();
      gw.ClrScr();
      var d = new Date();
      var dateText = "";
      var timeText = "";
    gw.SetBacklight(2,60); //one minut initially
      var type = -1;


      days = new Array("Sunday", "Monday", "Tuesday", "Wednesday",
                   "Thursday", "Friday", "Saturday");
      months = new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                     "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");

      do
      {
            // Display text
            gw.ShowImage(TRIPLEUP, "Brand");
            //Get time.
            var time = gw.GetTime();
            d.setTime(time * 1000);

            //Date.
            //text = d.toDateString();
            var text = days[d.getDay()] + " " +
                d.getDate() + " " + months[d.getMonth()] + " " +
d.getFullYear();
            if (text != dateText)
            {
                  dateText = text;
                  gw.ShowTextL(TRIPLELOW, dateText);
            }

            //Time.
            var hour = d.getHours();
            var minute = d.getMinutes();
            if (minute <= 9)
                  text = hour + ":0" + minute;
            else
                text = hour + ":" + minute;
            if (text != timeText)
            {
                  timeText = text;
                  gw.ShowTextL(TRIPLEMID + BIGFNT, timeText);
            }

                // Check if the last event was a buttonpress
                if ((type == gw.SEvent()) || (type == gw.LEvent()) || (type
== gw.REvent()))
                {
                        gw.SetBacklight(2,10);
                }

            //Set timeout.
            gw.SetTimeout(1, 30);

                // Activate all buttons
                gw.ActivateButtons(0x1f);

      } while ((type = gw.WaitForEvent(resp)) != gw.EXITEvent()&&type !=
gw.IEvent());

        gw.SetBacklight(2,0);

        if (type == gw.IEvent()) //launch Info script
        {
            var Par = new Array();
            gw.ActivateScript("Info", 1, Par, 0, "Normal");
        }
        gw.ClrTimeout(1);
}
//Execute script.
exitEventDetected = 0;
main();
```

11 Web server integration

This chapter describes how to configure a web server and build an application for data collection. It is also described how to use the sample test server.

11.1 Getting started

Before any configuration is initiated, make sure the following is available:

Hardware:

- A computer with Internet connection and at least one serial port (stand alone or connected to the local area network).
- RTX337x

Software:[2]

- Microsoft Windows 2003
- ServerMicrosoft Visual Studio 2005
- Microsoft .NET development kit
- OpenSSL
- The RTX337x server package from the eSupport site
- Cygwin

Other:

- An ISP account with phone number, username and password

Software available on the eSupport site:

- RTX337x-SERVER-x_x.zip (Sample Application Software)

Manuals available on the eSupport site:

- Technical reference manual (this document)

11.2 Step by Step configuration

This “Step by Step” section describes in details how to prepare a computer for use as a web server and which services must be enabled, in order to be able to collect data from the RTX337x. The description is based on a server with:

- MS Windows Server 2003 operating system.
- IIS 6.0 (Internet Information Services).
- MS Visual Studio 2005.

NOTE: If MS Windows Server 2008 is used, Auto-tuning must be disabled using the *netsh interface tcp set global autotuning=disabled* command.

[2] The user of the software must own all Licenses needed. Tunstall Healthcare A/S renounces any obligation of installing or using third party software.

11.2.1 Upgrade the OS with SP1

Visual Studio 2005 requires Service Pack 1 to be installed. The Service Pack can be downloaded from www.microsoft.com/downloads.

11.2.2 Install MS Visual Studio 2005

Install the MS Visual Studio 2005 Professional Edition. The installation program also installs and enables Microsoft .NET Framework 2.0.

11.2.3 Install WSE 3.0

Install Web Services Enhancements 3.0 for Microsoft .NET. Be sure to select the option that installs Visual Studio tools. This enhancements packet enables SOAP to use MTOM attachments (large amounts of binary data). WSE 3.0 can be downloaded from www.microsoft.com/downloads.

11.2.4 Change IIS Metabase settings

In order to use SSL, some default IIS Metabase settings must be changed. The Microsoft Metabase Explorer (MBExplorer.exe) can be used to change the settings. It's a part of IIS 6.0 Resource Kit Tool, which can be downloaded here: www.microsoft.com/downloads.

- ConnectionTimeout must be changed to (minimum) 300 (sec.)
- UpLoadReadAheadSize must be changed to (minimum) 500000

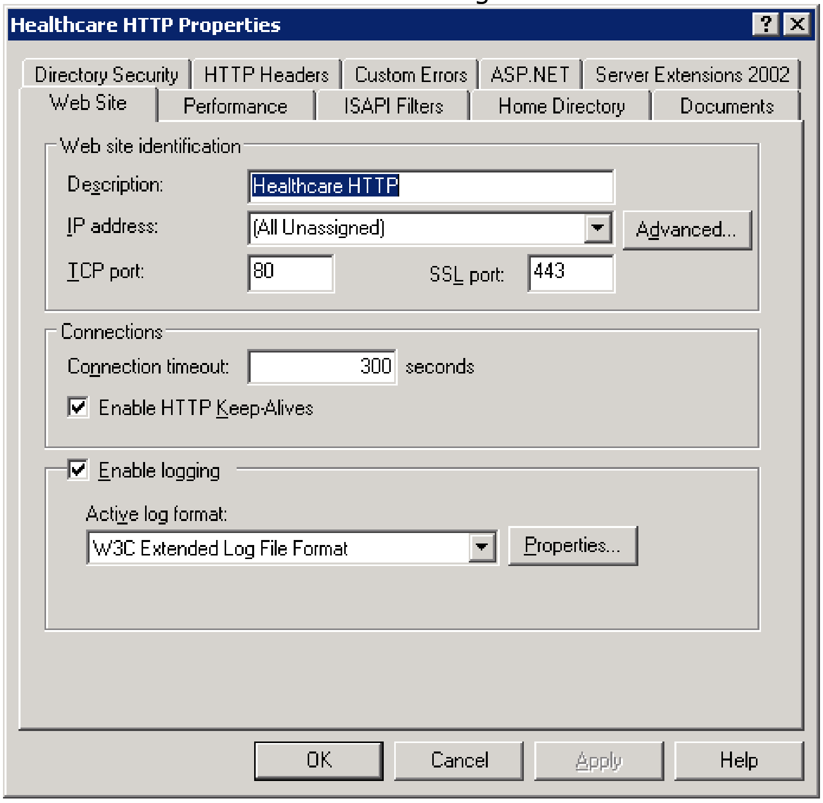
11.2.5 Create Folders (Names and paths are examples)

Create the folder e:\WEB\RTX337X and copy the content from the eSupport (server integration) to this folder. Change the e:\WEB\RTX337X\Data folder **Property->Security** by adding Users to "Groups or user names" and give Modify and Write permission.

11.2.6 Configure IIS

Open the IIS Manager (Start->Programs->Administrative Tools->Internet Information Services (IIS Manager). Create a Healthcare HTTP subfolder to the Web Sites folder. Use the **New->Web Site** Wizard to create the folder.

In the **Web Site** tab use this setting:



Web site identification

Description: Healthcare HTTP

IP address: (All Unassigned)

TCP port: 80 SSL port: 443

Connections

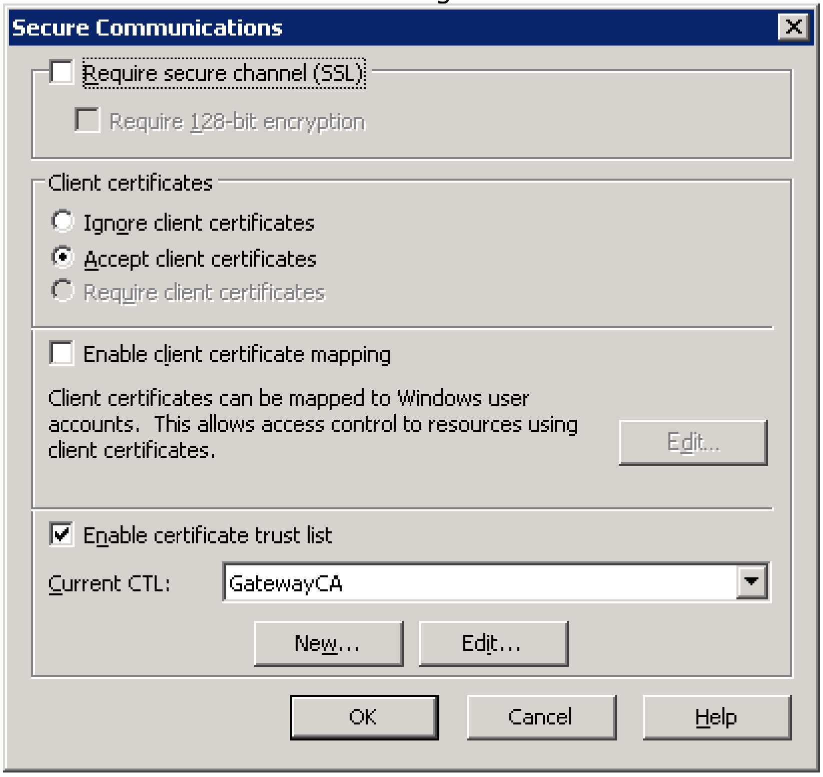
Connection timeout: 300 seconds

☑ Enable HTTP Keep-Alives

☑ Enable logging

Active log format: W3C Extended Log File Format

In the **Directory Security** tab run the Server Certificate wizard to enable SSL for the folder and subfolders. The settings should look like this:



☐ Require secure channel (SSL)

☐ Require 128-bit encryption

Client certificates

○ Ignore client certificates

● Accept client certificates

○ Require client certificates

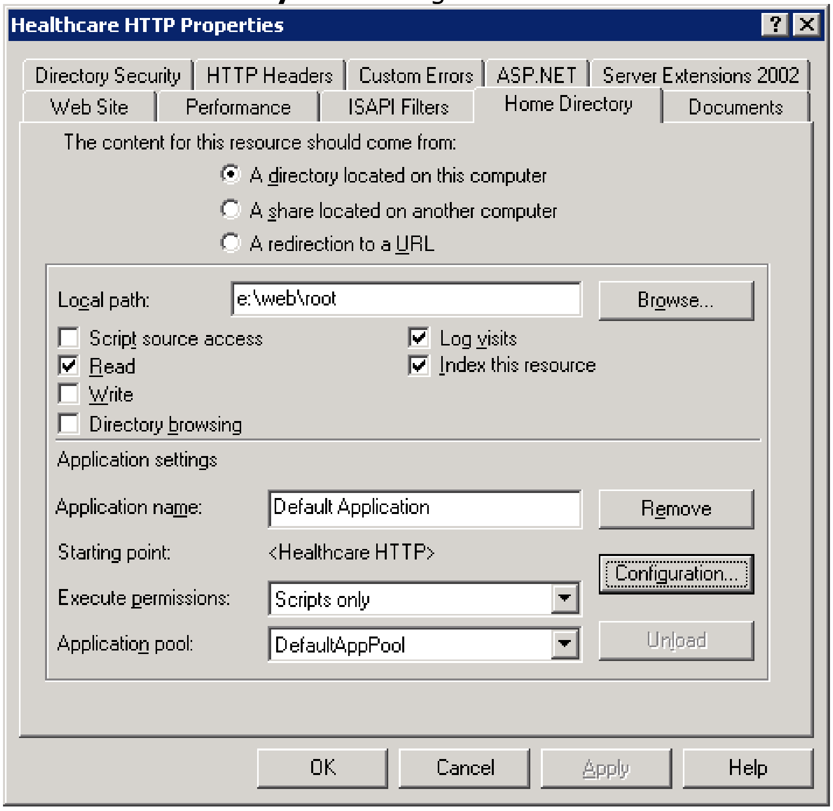
☐ Enable client certificate mapping

Client certificates can be mapped to Windows user accounts. This allows access control to resources using client certificates.

☑ Enable certificate trust list

Current CTL: GatewayCA

The **Home Directory** tab settings are:



The content for this resource should come from:

● A directory located on this computer
○ A share located on another computer
○ A redirection to a URL

Local path: e:\web\root

□ Script source access
☑ Read
□ Write
□ Directory browsing
☑ Log visits
☑ Index this resource

Application settings

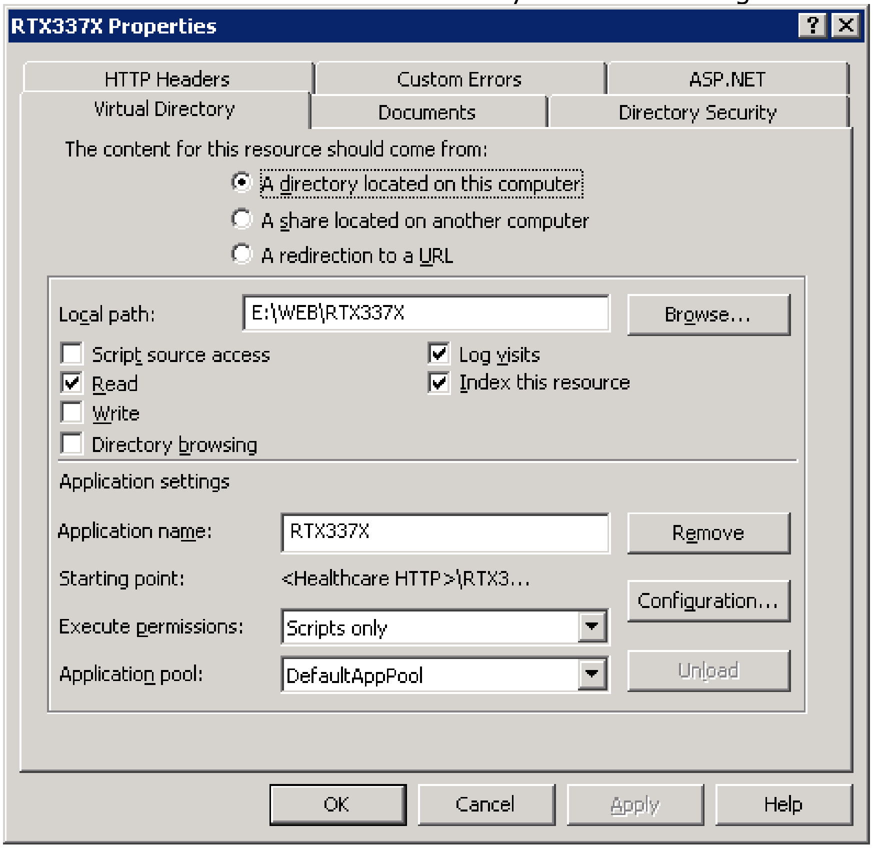
Application name: Default Application

Starting point: <Healthcare HTTP>

Execute permissions: Scripts only

Application pool: DefaultAppPool

Right click on the Healthcare HTTP folder and use the **New->Virtual Directory** wizard to create the RTX337X virtual directory with this setting:



The content for this resource should come from:

● A directory located on this computer
○ A share located on another computer
○ A redirection to a URL

Local path: E:\WEB\RTX337X

□ Script source access
☑ Read
□ Write
□ Directory browsing
☑ Log visits
☑ Index this resource

Application settings

Application name: RTX337X

Starting point: <Healthcare HTTP>\RTX3...

Execute permissions: Scripts only

Application pool: DefaultAppPool

11.2.7 SSL Server Configuration

Two certificates must be installed on the server in order to use SSL:

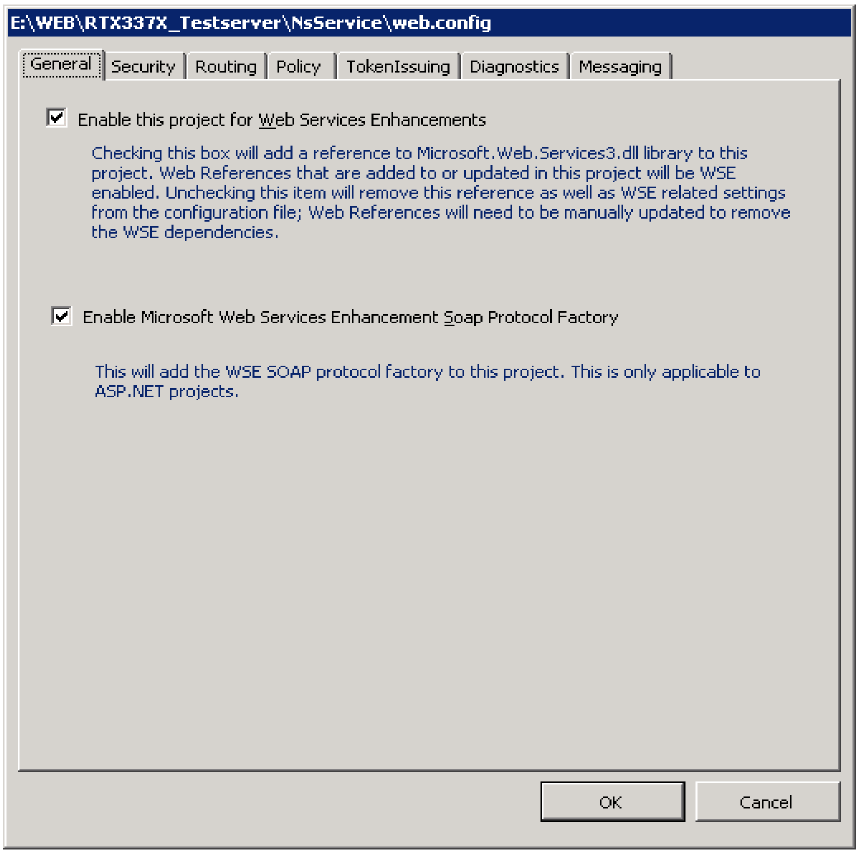
- Server.pfx
- Ca.crt

The certificates are created by openssl, see the details in section SSL of how to generate and install the certificates on the server.

11.2.8 Start a new Web Service

In order to start a new web service instead of using the test application from the RTX337x eSupport site:

- Generate interface source from the RTX33XX.wsdl file by using the wsdl.exe tool with the /si option.
- In Visual Studio create a new project: New->Web Site->ASP.NET Web Service
- Add the wsdl.exe generated file to the project.
- In Solution explorer: Right click project name. Click WSE Settings 3.0 and make these settings**:**



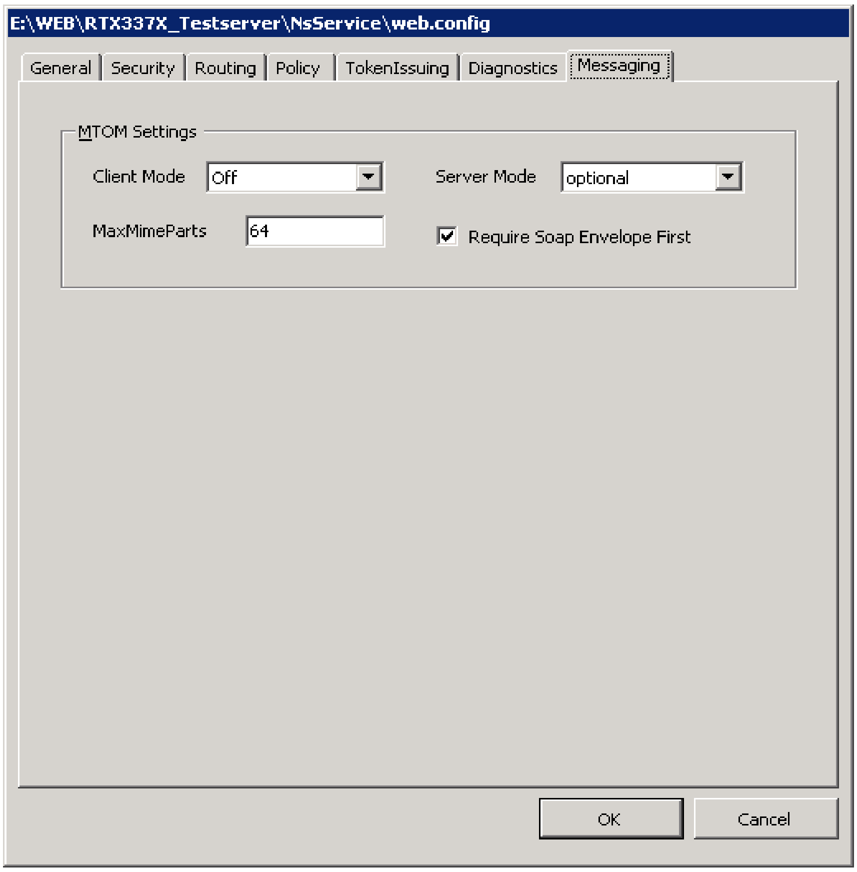
E:\WEB\RTX337X_Testserver\NsService\web.config

☑ Enable this project for Web Services Enhancements

Checking this box will add a reference to Microsoft.Web.Services3.dll library to this project. Web References that are added to or updated in this project will be WSE enabled. Unchecking this item will remove this reference as well as WSE related settings from the configuration file; Web References will need to be manually updated to remove the WSE dependencies.

☑ Enable Microsoft Web Services Enhancement Soap Protocol Factory

This will add the WSE SOAP protocol factory to this project. This is only applicable to ASP.NET projects.



E:\WEB\RTX337X_Testserver\NsService\web.config

MTOM Settings

Client Mode: Off

Server Mode: optional

MaxMimeParts: 64

☑ Require Soap Envelope First

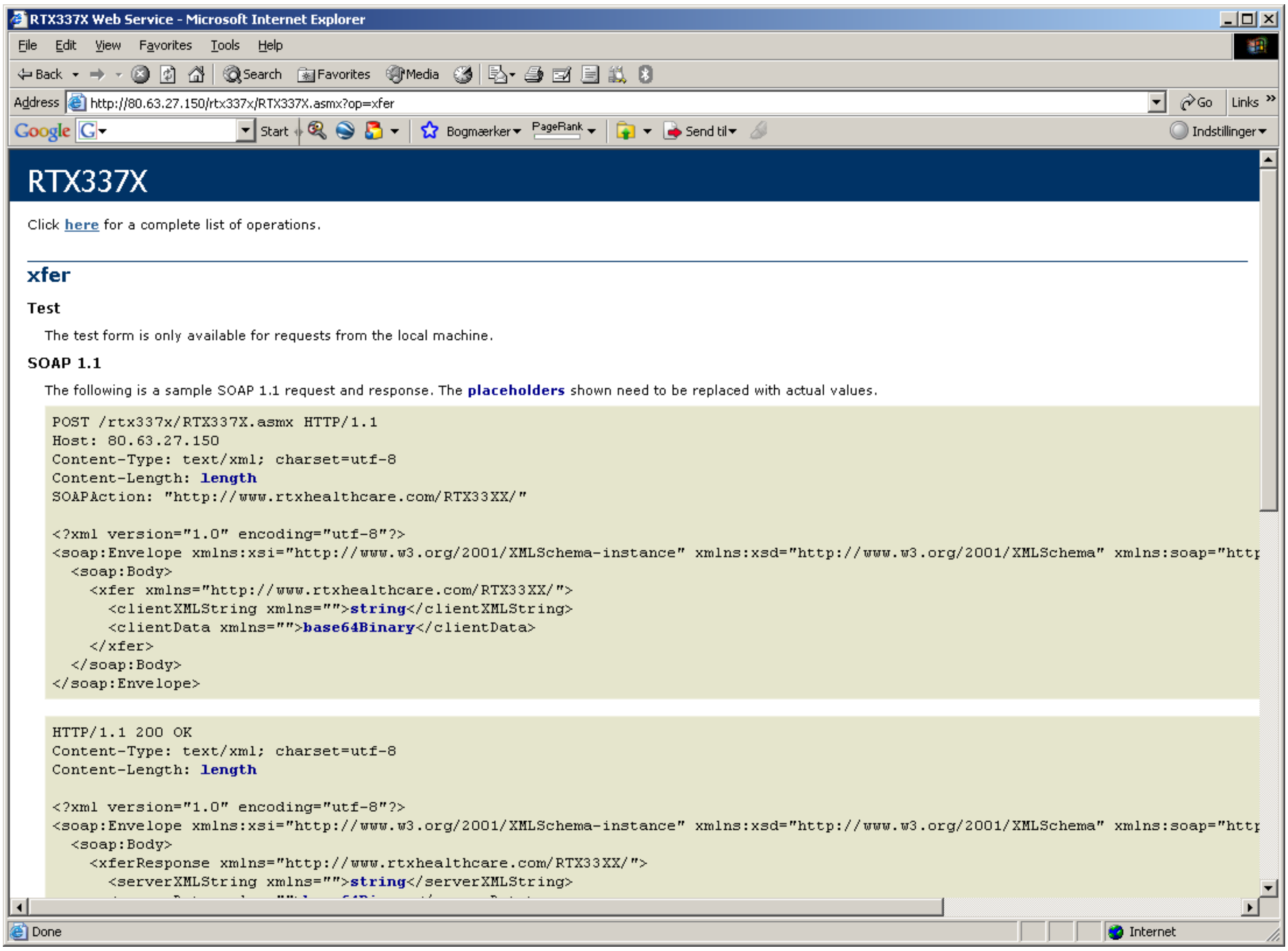
11.3 Sample application project

To get started with data collection easily, a simple sample application project is enclosed on the RTX337x eSupport site.
This project can be opened from Microsoft Visual Studio .NET for development purpose.

The server package on the eSupport site contains the files needed to receive data from the RTX337x, e.g measurement data from a blood pressure measurement or a weight measurement.

Follow the instructions:

- Unzip the RTX337x-SERVER-x_x.zip file from the eSupport site and copy the content of the Sample Application Software to the e:\WEB\RTX337X folder.
- Try to reach the folder from Internet Explorer with the command: HTTP://<Your web servers address>/RTX337X/RTX337X.asmx?op=xfer


RTX337X Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Google Start Bogmærker PageRank Send til Indstillinger
Done
Internet

Address: http://80.63.27.150/rtx337x/RTX337X.asmx?op=xfer

RTX337X

Click here for a complete list of operations.

xfer

Test

The test form is only available for requests from the local machine.

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /rtx337x/RTX337X.asmx HTTP/1.1
Host: 80.63.27.150
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.rtxhealthcare.com/RTX33XX/"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http
  <soap:Body>
    <xfer xmlns="http://www.rtxhealthcare.com/RTX33XX/">
      <clientXMLString xmlns="">string</clientXMLString>
      <clientData xmlns="">base64Binary</clientData>
    </xfer>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http
  <soap:Body>
    <xferResponse xmlns="http://www.rtxhealthcare.com/RTX33XX/">
      <serverXMLString xmlns="">string</serverXMLString>
```

If this screen is shown, the configuration is correct, and the web service is reachable from the RTX337x (if WSPATH is set to RTX337X/RTX337x.asmx and WS is set to the web server's global ip address).

11.4 Getting data on the web Server

The section describes how to create the application for receiving data from the RTX337x using the sample test server.

In the e:\WEB\RTX337X folder doubleclick RTX337X.sln to start the application in the Visual Studio Development Environment.

The main file in the application is RTX337X.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;
using System.Text;
using System.Web.Services;
using System.Web.Services.Protocols;

using System.Xml;
using System.Xml.Schema;
using System.Xml.Serialization;
using System.Xml.Xsl;


/// <summary>
/// Summary description for RTX337X
/// </summary>


[WebService(Namespace = "http://www.rtxhealthcare.com/RTX337X/")]
public class RTX337X : IRTX33XXSoap
{
    string GatewayIdTag = "";
    string TimeTag = "";
    string MessageNumberTag = "";
    string DvTypeTag = "";
    string GwInfoTag = "";
    string GwStatusTag = "";
    string GwChksumTag = "";
    string DeviceIdTag = "";
    string TypeTag = "";
    string UtcTimeTag = "";
    string BatteryStatusTag = "";
    string TimeDiffTag = "";
    string ValueTag = "";
    string DvChksumTag = "";
    string serverXMLString = "";
    XmlDocument Cdoc;
    XmlDocument Sdoc;
    string ClientXmlErr = "";
    string ServerXmlErr = "";
    string ActiveXSD = "";

    string timeformat = "dd/MM/yyyy HH.mm.ss";

    static string ProjectPath = new
DirectoryInfo(AppDomain.CurrentDomain.BaseDirectory).Parent.FullName;
    string DataPath = ProjectPath + "\\Data\\";
    string ServicePath = ProjectPath + "\\NsService\\";
    string ClientXSD = "";
    string ServerXSD = "";
    string CXmlNs = "";
    string SXmlNs = "";

    string XMLEncoding = "";

    public RTX337X()
    {
        //
        // TODO: Add constructor logic here
        //
    }

    [WebMethod]
    public string xfer(string clientXMLString, byte[] clientData, out byte[] serverData, out
bool ret)
    {
        clientXMLString = clientXMLString.Replace("{", "\r");
        clientXMLString = clientXMLString.Replace("}", "\n");

        // Find XML encoding
        if (clientXMLString.Contains("iso-8859-1"))
            XMLEncoding = "ISO-8859-1";
        else
            XMLEncoding = "UTF-8";
        FindNamespaces(clientXMLString);
        EnsureDirectory(new DirectoryInfo(DataPath + GetTagContent("GatewayId",
clientXMLString)));

        Cdoc = new XmlDocument();
        Sdoc = new XmlDocument();

        // Initialize empty server data
        serverData = new byte[4];
        for (int i = 0; i < 4; i++)
            serverData[i] = (byte)0;

        //Initialize Server XML
        CreateDefaultXml(Sdoc, SXmlNs);
        InsertValue(Sdoc, "Monitor", "Type", "NONE");

        if (ValidateXML(ClientXSD, ref clientXMLString))
        {

            Cdoc.LoadXml(clientXMLString);

            // Read content of client string
            GatewayIdTag = GetValue(Cdoc, "Gateway", "GatewayId");
            TimeTag = GetValue(Cdoc, "Gateway", "Time");
            MessageNumberTag = GetValue(Cdoc, "Gateway", "MessageNumber");
            DvTypeTag = GetValue(Cdoc, "Gateway", "DvType");
            GwInfoTag = GetValue(Cdoc, "Gateway", "GwInfo");
            GwStatusTag = GetValue(Cdoc, "Gateway", "GwStatus");
            GwChksumTag = GetValue(Cdoc, "Gateway", "GwChksum");
            DeviceIdTag = GetValue(Cdoc, "Monitor", "DeviceId");
            TypeTag = GetValue(Cdoc, "Monitor", "Type");
            UtcTimeTag = GetValue(Cdoc, "Monitor", "UtcTime");
            BatteryStatusTag = GetValue(Cdoc, "Monitor", "BatteryStatus");
            TimeDiffTag = GetValue(Cdoc, "Monitor", "TimeDiff");
            ValueTag = GetValue(Cdoc, "Monitor", "Value");
            DvChksumTag = GetValue(Cdoc, "Monitor", "DvChksum");

            InsertValue(Sdoc, "Gateway", "GatewayId", GatewayIdTag);

            // Replace unconverted ">" characters
            ValueTag = ValueTag.Replace("&gt;", ">");

            // Check checksum
            if (CheckChecksum())
            {
                ret = true;
                // Process message
                switch (TypeTag)
                {
                    // A request from the client to be initialized with configuration data
                    // for factory testing. The data is sent on the next REQUEST
                    case "REQUESTFACTORYTEST":
                        InsertValue(Sdoc, "Monitor", "Type", "NONE");
                        SetRequestReturn(GatewayIdTag, "REQUESTFACTORYTEST");
                        break;

                    // A request from the client to be initialized with configuration data
                    // for normal use. Can be partial or full patient initialization depending
                    // on the content of ValueTag. The data is sent on the next REQUEST
                    case "REQUESTPATIENTINITIALIZATION":
                        InsertValue(Sdoc, "Monitor", "Type", "NONE");
                        SetRequestReturn(GatewayIdTag, "REQUESTPATIENTINITIALIZATION");
                        break;

                    // A request from the client to the server for actions to be done by the
                    // client. The response is SENDLOG, CONFIG or NONE
                    case "REQUEST":
                        InsertValue(Sdoc, "Monitor", "Type", GetRequestReturn(GatewayIdTag,
Sdoc, ref serverData));
                        break;

                    // A request from the client telling the server, that the client wants a
                    // remote firmware download. On the next REQUEST from the client, the
server
```

```
                        // returns CONFIG in the TypeTag and the code as binary data
                        case "SENDRFU":
                            InsertValue(Sdoc, "Monitor", "Type", "NONE");
                            SetRequestReturn(GatewayIdTag, "CRFU");
                            break;

                        // No special action
                        default:
                            InsertValue(Sdoc, "Monitor", "Type", "NONE");
                            break;

                    }
                }
                else
                    ret = false;
            }
            else
            {
                Cdoc.LoadXml(clientXMLString);
                InsertValue(Sdoc, "Gateway", "GatewayId", GetValue(Cdoc, "Gateway", "GatewayId"));
                InsertValue(Sdoc, "Monitor", "Value", ClientXmlErr);
                ret = false;
            }

            SetTimeTag(Sdoc);
            SetChecksum(Sdoc);

            serverXMLString = Sdoc.OuterXml;
            if (!ValidateXML(ServerXSD, ref serverXMLString))
            {
                // Server XML failed validation
                Sdoc.LoadXml(serverXMLString);
                InsertValue(Sdoc, "Gateway", "GatewayId", GetValue(Cdoc, "Gateway", "GatewayId"));
                InsertValue(Sdoc, "Monitor", "Type", "NONE");
                InsertValue(Sdoc, "Monitor", "Value", ClientXmlErr + ServerXmlErr);
                SetChecksum(Sdoc);
                serverXMLString = Sdoc.OuterXml;
                ret = false;
            }
            serverXMLString = serverXMLString.Replace("\r", "{");
            serverXMLString = serverXMLString.Replace("\n", "}");
            CommLog(Cdoc, Sdoc, clientData);
            return serverXMLString;
        }

        private string GetValue(XmlDocument doc, string node, string tag)
        {
            XmlNamespaceManager nsm = new XmlNamespaceManager(doc.NameTable);
            nsm.AddNamespace("ns", doc.DocumentElement.NamespaceURI);
            node = "//ns:" + node;
            XmlNode root = doc.DocumentElement.SelectSingleNode(node, nsm);

            string Str = "";
            try
            {
                Str = root[tag].InnerText;
            }
            catch
            {
                Str = "";
            }
            return Str;
        }

        private uint adler32(uint adler, string Sbuf)
        {

            uint s1 = adler & 0xffff;
            uint s2 = (adler >> 16) & 0xffff;
            int n, len;
            Byte[] buf;

            Encoding encoding = Encoding.GetEncoding(XMLEncoding);
            buf = encoding.GetBytes(Sbuf);
            len = buf.Length;

            for (n = 0; n < len; n++)
            {
                s1 = (s1 + buf[n]) % 65521;
                s2 = (s2 + s1) % 65521;
            }
            return (s2 << 16) + s1;
        }

        private void SetChecksum(XmlDocument doc)
        {
            uint chksum;
            string str = "";

            // Set Gateway checksum
            chksum = 1;
            str = GetValue(doc, "Gateway", "GatewayId");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Gateway", "Time");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Gateway", "MessageNumber");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Gateway", "DvType");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Gateway", "GwInfo");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Gateway", "GwStatus");
            chksum = adler32(chksum, str);
            InsertValue(doc, "Gateway", "GwChksum", Convert.ToString(chksum));

            // Set Monitor checksum
            chksum = 1;
            str = GetValue(doc, "Monitor", "DeviceId");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Monitor", "Type");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Monitor", "UtcTime");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Monitor", "BatteryStatus");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Monitor", "TimeDiff");
            chksum = adler32(chksum, str);
            str = GetValue(doc, "Monitor", "Value");
            chksum = adler32(chksum, str);
            InsertValue(doc, "Monitor", "DvChksum", Convert.ToString(chksum));

        }

        private bool CheckChecksum()
        {
            bool ret = true;
            uint chksum;

            // Check Gateway checksum
            chksum = 1;
            chksum = adler32(chksum, GatewayIdTag);
            chksum = adler32(chksum, TimeTag);
            chksum = adler32(chksum, MessageNumberTag);
            chksum = adler32(chksum, DvTypeTag);
            chksum = adler32(chksum, GwInfoTag);
            chksum = adler32(chksum, GwStatusTag);

            try
            {
                if (chksum != Convert.ToUInt32(GwChksumTag))
                {
                    InsertValue(Sdoc, "Monitor", "Value", "Gateway checksum error");
                    ret = false;
                }
            }
            catch
            {
                InsertValue(Sdoc, "Monitor", "Value", "Gateway checksum error");
                return false;
            }

            // Check Monitor checksum
            chksum = 1;
            chksum = adler32(chksum, DeviceIdTag);
            chksum = adler32(chksum, TypeTag);
            chksum = adler32(chksum, UtcTimeTag);
            chksum = adler32(chksum, BatteryStatusTag);
            chksum = adler32(chksum, TimeDiffTag);
            chksum = adler32(chksum, ValueTag);

            try
            {
                if (chksum != Convert.ToUInt32(DvChksumTag))
                {
                    InsertValue(Sdoc, "Monitor", "Value", "Monitor checksum error");
                    ret = false;
                }
            }
            catch
            {
                InsertValue(Sdoc, "Monitor", "Value", "Monitor checksum error");
                return false;
            }

            return ret;
        }

        private void CreateDefaultXml(XmlDocument doc, string ns)
        {
            doc.LoadXml("<?xml version=\"1.0\" encoding=\"" + XMLEncoding + "\"?>" +
                "<xml xmlns=\"" + ns + "\">" +
                    "<Gateway>" +
                    "</Gateway>" +
                    "<Monitor>" +
                    "</Monitor>" +
                "</xml>");
        }

        private void InsertValue(XmlDocument doc, string node, string tag, string val)
        {
            XmlNamespaceManager nsm = new XmlNamespaceManager(doc.NameTable);
            nsm.AddNamespace("ns", doc.DocumentElement.NamespaceURI);
            node = "//ns:" + node;
            XmlNode root = doc.DocumentElement.SelectSingleNode(node, nsm);
            XmlElement elem = doc.CreateElement(tag, doc.DocumentElement.NamespaceURI);
            elem.InnerText = val;
            try
            {
                root.ReplaceChild(elem, root[tag]);
            }
            catch
            {
                root.AppendChild(elem);
            }
        }

        private void CommLog(XmlDocument Cdoc, XmlDocument Sdoc, byte[] clientdata)
        {
            string fname1 = DataPath + GetValue(Cdoc, "Gateway", "GatewayId") + "\\CommLog.txt";
            string fname2 = "";
            string ctime = "";
            string stime = "";
            DateTime Srv_t = DateTime.Now;

            // Save clientdata if not empty default array
            if ((clientdata != null) && (clientdata.Length > 4))
            {
                int cnt=1;
                FileInfo fi;
                for (;;)
                {
                    fname2 = DataPath + GetValue(Cdoc, "Gateway", "GatewayId") + "\\ClientData_" +
    cnt + ".bin";
                    fi = new FileInfo(fname2);
                    if (!fi.Exists)
                        break;
                    cnt++;
                }

                try
                {
                    BinaryWriter bw = new BinaryWriter(File.Open(fname2, FileMode.Create));
                    bw.Write(clientdata);
                    bw.Close();
                }
                catch
                {
                }
            }

            // Make time info strings
            try
            {
                ctime = "[Srv:" + Srv_t.ToString(timeformat);
                if (TimeTag.Length > 0)
                    ctime += "  <Time>:" + GetTimeString(TimeTag);
                if (UtcTimeTag.Length > 0)
                    ctime += " <UtcTime>:" + GetTimeString(UtcTimeTag);
                ctime += "]";
            }
            catch
            {
                ctime = "[Time format ERROR]";
            }

            try
            {
                stime = "[Srv:" + Srv_t.ToString(timeformat);
                if (GetValue(Sdoc, "Gateway", "Time").Length > 0)
                    stime += "  <Time>:" + GetTimeString(GetValue(Sdoc, "Gateway", "Time"));
                if (GetValue(Sdoc, "Monitor", "UtcTime").Length > 0)
                    stime += " <UtcTime>:" + GetTimeString(GetValue(Sdoc, "Monitor", "UtcTime"));
                stime += "]";
            }
            catch
            {
                stime = "[Time format ERROR]";
            }

            AddStringToFile(fname1, "[Client XML]" + ctime);
            AddXMLToFile(fname1, Cdoc);
            AddStringToFile(fname1, "\r\n");
            AddStringToFile(fname1, "[Server XML]" + stime);
            AddXMLToFile(fname1, Sdoc);
            AddStringToFile(fname1, "\r\n\r\n");

        }

        private void LoadFileData(String fname, XmlDocument Sdoc, ref byte[] serverData)
        {
            string tfname = fname + ".txt";
            string bfname = fname + ".bin";

            // Read XML data
            try
            {
                StreamReader sr = new StreamReader(tfname, Encoding.GetEncoding("utf-8"));
                InsertValue(Sdoc, "Monitor", "Value", sr.ReadToEnd());
                sr.Close();
            }
            catch (Exception)
            {
                InsertValue(Sdoc, "Monitor", "Value", "Unable to read " + tfname);
                return;
            }
```

```
            // Read binary data
            try
            {
                FileInfo fi = new FileInfo(bfname);
                long numBytes = fi.Length;
                Array.Resize(ref serverData, (int)numBytes);
                FileStream fs = new FileStream(bfname, FileMode.Open, FileAccess.Read);
                BinaryReader br = new BinaryReader(fs);
                serverData = br.ReadBytes((int)numBytes);
                br.Close();
                fs.Close();
            }
            catch (Exception)
            {
                // Do nothing. Binary data is not mandatory.
                //InsertValue(Sdoc, "Monitor", "Value", "Unable to read " + bfname);
            }
        }

        private string GetRequestReturn(String id, XmlDocument Sdoc, ref byte[] serverData)
        {
            string ret = "NONE";
            string str = "";
            int val = 0;
            string tfname = DataPath + id + "\\RequestReturn.txt";
            try
            {
                StreamReader sr = new StreamReader(tfname, Encoding.GetEncoding("utf-8"));
                str = sr.ReadToEnd();
                sr.Close();
                val = Convert.ToInt16(str);
            }
            catch (Exception)
            {
            }

            if ((val & 0x0001) != 0)
            {
                ret = "SENDLOG";
                val &= ~0x0001;
            }
            else if ((val & 0x0002) != 0)
            {
                LoadFileData(DataPath + id + "\\ConfigData", Sdoc, ref serverData);
                ret = "CONFIG";
                val &= ~0x0002;
            }
            else if ((val & 0x004) != 0)
            {
                LoadFileData(DataPath + id + "\\RFUData", Sdoc, ref serverData);
                ret = "CONFIG";
                val &= ~0x0004;
            }
            else if ((val & 0x008) != 0)
            {
                LoadFileData(DataPath + id + "\\FactoryTest", Sdoc, ref serverData);
                ret = "CONFIG";
                val &= ~0x0008;
            }

            else if ((val & 0x010) != 0)
            {
                LoadFileData(DataPath + id + "\\PatientInit", Sdoc, ref serverData);
                ret = "CONFIG";
                val &= ~0x0010;
            }

            try
            {
                StreamWriter sw = new StreamWriter(tfname, false, Encoding.GetEncoding("utf-8"));
                sw.Write(Convert.ToString(val));
                sw.Close();
            }
            catch (Exception)
            {
            }
            return ret;
        }

        private void SetRequestReturn(String id, String str)
        {
            int val = 0;
            string strval = "";
            string tfname = DataPath + id + "\\RequestReturn.txt";
            try
            {
                StreamReader sr = new StreamReader(tfname, Encoding.GetEncoding("utf-8"));
                strval = sr.ReadToEnd();
                sr.Close();
                val = Convert.ToInt16(strval);
            }
            catch (Exception)
            {
            }

            if (str.CompareTo("CRFU") == 0)
                val |= 0x0004;

            if (str.CompareTo("REQUESTFACTORYTEST") == 0)
                val |= 0x0008;

            if (str.CompareTo("REQUESTPATIENTINITIALIZATION") == 0)
                val |= 0x0010;

            try
            {
                StreamWriter sw = new StreamWriter(tfname, false, Encoding.GetEncoding("utf-8"));
                sw.Write(Convert.ToString(val));
                sw.Close();
            }
            catch (Exception)
            {
            }
        }

        private void SetTimeTag(XmlDocument doc)
        {
            TimeSpan t = (DateTime.Now - new DateTime(1970, 1, 1));
            int timestamp = (int)t.TotalSeconds;
            InsertValue(doc, "Gateway", "Time", timestamp.ToString());
        }

        private string GetTimeString(string val)
        {
            DateTime t = new DateTime(1970, 1, 1);
            string rtn = "";
            if (val.Length > 0)
            {
                t = t.AddSeconds(Convert.ToDouble(val));
                rtn = t.ToString(timeformat);
            }
            return rtn;
        }

        private void AddStringToFile(string fname, string str)
        {
            try
            {
                StreamWriter sw = new StreamWriter(fname, true, Encoding.GetEncoding("utf-8"));
                sw.WriteLine(str);
                sw.Close();
            }
            catch
            {
            }
        }
```

```
private void AddXMLToFile(string fname, XmlDocument doc)
{
    // Write the XML to a memory stream using the XmlTextWriter set to use indents
    // and utf-8 format
    MemoryStream ms = new MemoryStream();
    XmlTextWriter writer = new XmlTextWriter(ms, Encoding.GetEncoding("utf-8"));
    writer.Indentation = 3;
    writer.Formatting = Formatting.Indented;
    doc.Save(writer);
    String buf = Encoding.UTF8.GetString(ms.ToArray());
    writer.Close();

    // We always save the XML in utf-8 format, but the XmlTextWriter replaces the
    // original encoding information in the XML. The original information is re-inserted
    buf = buf.Replace("utf-8", XMLEncoding);

    // Add to logfile
    StreamWriter sw = new StreamWriter(fname, true);
    sw.Write(buf);
    sw.Close();
}

private bool ValidateXML(string xsd, ref string xml)
{


    ValidationEventHandler eventhandler = new ValidationEventHandler(ValidationCallback);
    XmlReaderSettings settings = new XmlReaderSettings();
    settings.ValidationType = ValidationType.Schema;
    try
    {
        settings.Schemas.Add(null, xsd);
    }
    catch
    {
        ClientXmlErr = "\r\n[XSD schema load error]\r\n";
        ClientXmlErr += "Unable to load " + xsd;
        return false;
    }
    settings.ValidationEventHandler += eventhandler;
    StringReader streamxml = new StringReader(xml);
    XmlReader reader = XmlReader.Create(streamxml, settings);
    ActiveXSD = xsd;
    try
    {
        while (reader.Read()) ;
    }
    catch
    {
        // Not parseable content in <Value> tag
        string OldValue = GetTagContent("Value", xml);
        if (OldValue.Length > 0)
        {
            string NewValue = OldValue.Replace("&", "&amp;");
            NewValue = NewValue.Replace(">", "&gt;");
            NewValue = NewValue.Replace("<", "&lt;");
            xml = xml.Replace(OldValue, NewValue);
            ClientXmlErr = "\r\n[Client XML error]\r\n";
            ClientXmlErr += "Not parseable characters in Value tag has been escaped";
        }
        else
        {
            string gwid = GetTagContent("GatewayId", xml);
            xml = "<?xml version=\"1.0\"?>";
            xml += "<xml>";
            xml += "<Gateway><GatewayId>" + gwid + "</GatewayId><Time></Time></Gateway>";
            xml += "<Monitor><Value>XML content not parseable</Value></Monitor>";
            xml += "</xml>";
            ClientXmlErr = "\r\n[Client XML error]\r\n";
            ClientXmlErr += "XML content not parseable";
        }
        return false;
    }
    if (((ActiveXSD == ClientXSD) && ClientXmlErr.Length == 0) ||
            ((ActiveXSD == ServerXSD) && ServerXmlErr.Length == 0))
        {
            return true;
        }
        else
            return false;
    }

    private void ValidationCallback(object sender, ValidationEventArgs vargs)
    {
        if (ActiveXSD == ClientXSD)
        {
            if (ClientXmlErr.Length == 0)
                ClientXmlErr = "\r\n[Client XML error]\r\n";
            ClientXmlErr += vargs.Message;
        }
        else if (ActiveXSD == ServerXSD)
        {
            if (ServerXmlErr.Length == 0)
                ServerXmlErr = "\r\n[Server XML error]\r\n";
            ServerXmlErr += vargs.Message;
        }
    }

    private string GetTagContent(string tag, string xml)
    {
        string StartTag = "<" + tag + ">";
        string EndTag = "</" + tag + ">";
        int StartIdx = xml.IndexOf(StartTag);
        int EndIdx = xml.IndexOf(EndTag);
        if ((StartIdx > 0) && (EndIdx > 0))
        {
            StartIdx += StartTag.Length;
            return xml.Substring(StartIdx, EndIdx - StartIdx);
        }
        else return "";
    }

    private void FindNamespaces(string xml)
    {
        string StartTag = "xmlns=\"";
        string EndTag = "\"";
        int StartIdx = xml.IndexOf(StartTag);
        int EndIdx = xml.IndexOf(EndTag, StartIdx + StartTag.Length);
        if ((StartIdx > 0) && (EndIdx > 0))
        {
            StartIdx += StartTag.Length;
            CXmlNs = xml.Substring(StartIdx, EndIdx - StartIdx);
            SXmlNs = CXmlNs.Replace("Client", "Server");
            ClientXSD = ServicePath + CXmlNs + ".xsd";
            ServerXSD = ServicePath + SXmlNs + ".xsd";
        }
    }

    private static void EnsureDirectory(DirectoryInfo oDirInfo)
    {
        if (!oDirInfo.Exists)
        {
            oDirInfo.Create();
        }
    }
}
```

The application interacts with text and binary files in the DataPath folder, and all received and sent XML telegrams are logged in the textfile <GatewayId>_CommLog.txt.

When receiving an XML telegram, the response is controlled by the binary value of the character saved in the <GatewayId>_RequestReturn.txt file:

(val & 0x01) = Sendlog
SENDLOG is returned in the <Type> tag.

(val & 0x02) = Config
The bit is set manually.
CONFIG is returned in the <Type> tag
ConfigData.txt file content is inserted inside the <Value> tag
ConfigData.bin file is returned as binary attachment (MTOM)

(val & 0x04) = RemoteFirmwareUpdate
The bit is automatically set when the server receives a SENDRFU type from the RTX337x.
CONFIG is returned in the <Type> tag
RFUData.txt file content is inserted inside the <Value> tag
RFUData.bin file is returned as binary attachment (MTOM)

(val & 0x08) = FactoryTest
The bit is automatically set when the server receives a REQUESTFACTORYTEST type from the RTX337x.
CONFIG is returned in the <Type> tag
FactoryTest.txt file content is inserted inside the <Value> tag
FactoryTest.bin file is returned as binary attachment (MTOM)

(val & 0x10) = PatientInitialization
The bit is automatically set when the server receives a REQUESTPATIENTINITIALIZATION type from the RTX337x.
CONFIG is returned in the <Type> tag
PatientInitialization.txt file content is inserted inside the <Value> tag
PatientInitialization.bin file is returned as binary attachment (MTOM)

11.5 Writing the application

SOAP (RPC) is used as the interface protocol. Only one procedure is defined for exchange of data between client (RTX337x) and server (this function is autogenerated by WSE 3.0 from the wsdl specification in section 12.1):

```
public string xfer(string clientXMLString, byte[] clientData, out
byte[] serverData, out bool ret)
```

IMPORTANT:
SOAP does not support transmission of <CR> and <LF>. These characters are therefore converted to "{" and "}" respectively before sending the client string. They must be converted back to <CR> and <LF> before using the string. In the return server string, the server must do the same conversion.
The clientXMLSstring is the XML-telegram sent from the client to the server. The clientData is the binary data from the RTX337x. The serverData is the binary data from the server. The ret value is indicating if the received parameters has passed checksum test (OK = true). The return string is the serverXMLString.
The format of the XML data in the string is in detail described in chapter 12, XML.

11.6 GZIP

The RTX337x supports gzip compressed SOAP telegrams (from Firmware RTX337x-10_1 and up), both sending them through usage of the AT+pGZIPCOMP command (see section 9.2.35) and receiving them. The firmware's which support receiving gzip encoded contents sets the "Accept-Encoding" header to "gzip, deflate" to indicate this to the web server allowing it an easy way to decide whether to compress the response or not.

Incoming SOAP telegram encoding on the RTX337X is auto detected through looking at the HTTP "Encoding" header having the value 'gzip'. It is possible to transmit compressed SOAP telegrams to the RTX337X even though it is not enabled on the RTX337X transmitting data.

NOTE: Not all web servers support receiving and seamlessly decompress input HTTP POST (SOAP) data, but most should allow to selectively compress output if the client indicates it supports it.

11.7 SSL

To transmit secure patient data on the internet, SSL (Secure Socket Layer) is used.

> Note: **The OpenSSL and Cygwin programs must be installed for creating SSL certificates.**
> **The RTX337x must be configured for SSL.**

11.7.1 Creating and signing certificates (example)

Step #	Description
1	Create a self-signed Certificate Authority (CA) key and certificate: YourPrompt> openssl genrsa -des3 -out ca.key 1024. Generating RSA private key, 1024 bit long modulus ....++++++ ....++++++ e is 65537 (0x10001) Enter PEM pass phrase: password Verifying password - Enter PEM pass phrase: password YourPrompt>
2	Use the CA key to create a self-signed certificate: YourPrompt> openssl req -new -x509 -days 3650 -key ca.key -out ca.crt Using configuration from /etc/ssl/openssl.cnf Enter PEM pass phrase: password You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank For some fields there will be a default value, If you enter '.', the field will be left blank. ----- Country Name (2 letter code) [AU]: DK State or Province Name (full name) [Some-State]:. Locality Name (eg, city) []:Noerresundby Organization Name (eg, company) [Internet Widgits Pty Ltd]:Tunstall Healthcare Organizational Unit Name (eg, section) []:. Common Name (eg, YOUR name) []:CA Email Address []:. YourPrompt> This will give you a CA key (ca.key) and a certificate (ca.crt). Keep backups of both of these and Password in a secure location.
3	Create a server key for the server/service to secure: YourPrompt> openssl genrsa -des3 -out server.key 1024 Generating RSA private key, 1024 bit long modulus ....+++ ....+++ e is 65537 (0x10001) Enter PEM pass phrase: password Verifying password - Enter PEM pass phrase: password YourPrompt>
4	Generate a certificate request for the server key: YourPrompt> openssl req -new -key server.key -out server.csr

	You are about to be asked to enter information that will be incorporated into your certificate request. What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank For some fields there will be a default value, If you enter '.', the field will be left blank. ----- Country Name (2 letter code) [AU]:DK State or Province Name (full name) [Some-State]:. Locality Name (eg, city) []:Noerresundby Organization Name (eg, company) [Internet Widgits Pty Ltd]:Tunstall Healthcare Organizational Unit Name (eg, section) []:. Common Name (eg, YOUR name) []:80.63.27.150 *(Your servers global IP address or host name here)* Email Address []:. Please enter the following 'extra' attributes to be sent with your certificate request A challenge password []:. An optional company name []:. YourPrompt> A key part of this step is that the "Common Name (eg, YOUR name)" above must be the DNS name of the server as it will be seen from the client.
5	Sign the certificate signature request with your CA key: YourPrompt> sign.sh server.csr (included on the CD-ROM) CA signing: my_imap.csr -> my_imap.crt: Using configuration from ca.config Enter PEM pass phrase:password Check that the request matches the signature Signature ok The Subjects Distinguished Name is as follows countryName :PRINTABLE:'DK' stateOrProvinceName :PRINTABLE:'' localityName :PRINTABLE:'Noerresundby' organizationName :PRINTABLE:'RTX' organizationalUnitName:PRINTABLE:'' commonName :PRINTABLE:'Healthcare.RTX.net' Certificate is to be certified until Jan 20 21:35:12 2003 GMT (365 days) Sign the certificate? [y/n]:y 1 out of 1 certificate requests certified, commit? [y/n]y Write out database with 1 new entries Data Base Updated CA verifying: server.crt <-> CA cert server.crt: OK This will give you a server key (server.key) a server certificate request that will no longer be used (server.csr) and a server certificate (server.crt).
6	(NOTE:This section is not always necessary) Create Diffie-Hellman parameters with the following command: YourPrompt> openssl gendh -out server.dh 1024 Generating DH parameters, 1024 bit long safe prime, generator 2 This is going to take a long time ....+...+. [... it wasn't kidding about taking a long time, it goes on and on like this ...]

	...............................++*++*++* This will give you server DH parameters (server.dh).
7	Create a version of the server RSA key with no password: YourPrompt> openssl rsa –in server.key -out server_nopw.key read RSA key Enter PEM pass phrase: Writing RSA key
8	Create a PEM file called my_imap.pem containing pieces from three other files as follows: -----BEGIN RSA PRIVATE KEY----- [encoded key] -----END RSA PRIVATE KEY----- [empty line] -----BEGIN CERTIFICATE----- [encoded certificate] -----END CERTIFICATE----- [empty line] -----BEGIN DH PARAMETERS----- [encoded key] -----END DH PARAMETERS----- The RSA private key portion is in server_nopw.key. (For the client certificate use the key with password) The Certificate portion is in server.crt. (There is a bunch of extra junk in this file that isn't used, only include the part from ----BEGIN to -----END inclusive. The DH Parameters portion is in server.dh (not always used).

11.7.2 Client support

Finally, the CA certificate must be installed on each client machine. That can be done on Windows 2000 and Windows XP with these steps:

Step #	Description
1	Copy the ca.crt file to ca.cer on the Windows machine.
2	Right click it, pick "Install Certificate".
3	Click "Next".
4	Click "Place all certificates in the following store"
5	Click "Browse..."
6	Click "Trusted Root Certification Authorities" and click "Ok".
7	Click "Next".
8	Click "Finish".
9	Click "Yes".
10	Click "Ok".

Above explanation is for server authentication. For client authentication create a PEM file as described above in 3, 4, 5, 6, 7, 8, use user instead of server (user.crt etc.). Instead of common name (your server’s global IP address) use a name of your own choice. Then

```
YourPrompt> openssl pkcs12 –export –in user.pem –out user.pfx –name “RTX”
```

11.7.3 RTX337x configuration for SSL

Some settings has to be changed on the RTX337x in order to use SSL in the server communication:

```
AT+pSAUT=1<CR>
```

Enables server authentication.

```
AT+pCAUT=1<CR>
```

Enables client authentication. Requires the server authentication to be enabled also. If client authentication is not enabled, the client certificate (AT+pCCRT) doesn't need to be installed.

```
AT+pCACRT=<LF>
-----BEGIN CERTIFICATE-----<LF>
MIIC9zCCAmCgAwIBAgIBADANBgkqhkiG9w0BAQQFADBhMQswCQYDVQQGEwJESzET<LF>
MBEGA1UECBMKU29tZS1TdGF0ZTETMBEGA1UEBxMKTnIuIFN1bmRieTEbMBkGA1UE<LF>
ChMSUlRYIEhlYWx0aGNhcmUgQS9TMQswCQYDVQQDEwJDQTAeFw0wMzEwMDExMTE0<LF>
.
.
The ca.crt created in section 11.7.1 Step #2.
.
.
9w0BAQQFAAOBgQBviWf7wDBhfQ5TbgnggGBEqqc+SOv8jCn4EsBSJ9C+d1xgPZhT<LF>
P4rLa9s0jbHA7Xj3nCgyOjoBxBRnUWjcrDSHnabuNaw0IwJUIK2rgOugaYNL8qAv<LF>
LZaH+0EqhFs12TZ695xfrtc87zT3V8/ajtbQqAmVEdfTpAMENrJEtepKRA==<LF>
-----END CERTIFICATE-----<CR>
```

Installs the CA certificate.

NOTE: **OpenSSL generates the certificate with <CR><LF> as linebreaks. This is not allowed in the AT+pCACRT command as it will cause a premature command termination. Remove the <CR>'s as shown above.**
Be aware that some text editors automatically inserts <CR>'s at linebreaks.

```
AT+pCCRT=<LF>
-----BEGIN RSA PRIVATE KEY-----<LF>
Proc-Type: 4,ENCRYPTED<LF>
DEK-Info: DES-EDE3-CBC,08D5B318B87B8BB4<LF>
<LF>
GKgWtorZKRnglQfndyPTjUvpDzwbWoFIG3bZjwD3x1Bo3p7WfD8qGw1PEMQZlpph<LF>
fJ2kUvFGW06CMH0MCIbM9NclnIJAFdx+nBFdXG2H8wOjnWvmzM+14Cec4YeffoDO<LF>
RelHXTFHq5YB1ltTGS+RGdl5xUBIAfVkUXPZXM+G6zjt+famS89AHM2SHKgwrSPi<LF>
.
.
The user.key created in section 11.7.1 Step #3.
.
.
8ezRYjhcLYLI6aZNA9wBO9OHIH5pLouFQJ1ycWBr1rJjxvXgflIPPxKkH92esIV6<LF>
TK4NvQ5P1V85h0HnyHzgncoI56TzmNAICtgnEk65X7PdSL5cgwobHWIQpMgXlMv7<LF>
+y+NRMB9IrpZq01yc/o/ODCYOaLrYh9zGPY1uJRdMb2Z4kgjaG3igg==<LF>
-----END RSA PRIVATE KEY-----<LF>
-----BEGIN CERTIFICATE-----<LF>
MIICGzCCAYQCAQQwDQYJKoZIhvcNAQEEBQAwYTELMAkGA1UEBhMCREsxEzARBgNV<LF>
BAgTClNvbWUtU3RhdGUxEzARBgNVBAcTCk5yLiBTdW5kYnkxGzAZBgNVBAoTElJU<LF>
WCBIZWFsdGhjYXJlIEEvUzELMAkGA1UEAxMCQ0EwHhcNMDcwNTIzMTMyNTEzWhcN<LF>
.
```

```
.
The user.crt created in section 11.7.1 Step #5.
.
.
mteBbdkY6OQpjc+sIVIobcghH3rcOB7Gk/suibcBoiP+ilRvjU97+8dNMT2u2Kim<LF>
06SQCpj2aFEe+DLZ3aZBBlYvTa1LT6WiF6t04yKYuIUdTWg99fHQPZuI5hAATock<LF>
vBTJjNzrD+m7atzgZLbx<LF>
-----END CERTIFICATE-----<CR>
```

Installs the client certificate.

NOTE: **OpenSSL generates the certificates with <CR><LF> as linebreaks. This is not allowed in the AT+pCCCRT command as it will cause a premature command termination. Remove the <CR>’s as shown above.**
Be aware that some text editors automatically inserts <CR>’s at linebreaks.

11.7.4 Server configuration for SSL

Create a server.pfx from the server.pem:

YourPrompt> openssl pkcs12 –export –in server.pem –out server.pfx –name “RTX”

Copy server.pfx and ca.crt to the IIS 6.0 web server.

In order to view the Certificates store on the web server, perform the following steps:

- Click **Start**, and then click **Run**.
- Type "MMC.EXE" (without the quotation marks) and click **OK**.
- Click **Console** in the new MMC you created, and then click **Add/Remove Snap-in**.
- In the new window, click **Add**.
- Highlight the **Certificates** snap-in, and then click **Add**.
- Choose the **Computer** option and click **Next**.
- Select **Local Computer** on the next screen, and then click **OK**.
- Click **Close** , and then click **OK**.

You have now added the Certificates snap-in, which will allow you to work with any certificates in your computer's certificate store. You may want to save this MMC for later use.
Now that you have access to the Certificates snap-in, you can import the server certificate into you computer's certificate store by following these steps:
Open the Certificates (Local Computer) snap-in and navigate to **Personal**, and then **Certificates**.

Note: Certificates may not be listed. If it is not, that is because there are no certificates installed.
Right-click **Certificates** (or **Personal** if that option does not exist.)
Choose **All Tasks**, and then click **Import**.
When the wizard starts, click **Next**. Browse to the PFX file you created containing your server certificate and private key. Click **Next**.
Enter the password you gave the PFX file when you created it. Be sure the **Mark the key as exportable** option is selected if you want to be able to export the key pair again from this computer. As an added security measure, you may want to leave this option unchecked to ensure that no one can make a backup of your private key.
Click **Next**, and then choose the Certificate Store you want to save the certificate to. You should select **Personal** because it is a Web server certificate. If you included the certificates in the certification hierarchy, it will also be added to this store.

Click **Next**. You should see a summary of screen showing what the wizard is about to do. If this information is correct, click **Finish**.
You will now see the server certificate for your Web server in the list of Personal Certificates. It will be denoted by the common name of the server (found in the subject section of the certificate).
Now that you have the certificate backup imported into the certificate store, you can enable Internet Information Services 6.0 to use that certificate (and the corresponding private key). To do this, perform the following steps:
Open the Internet Services Manager (under Administrative Tools) and navigate to the Web site you want to enable secure communications (SSL/TLS) on.
Right-click on the site and click **Properties**.
You should now see the properties screen for the Web site. Click the **Directory Security** tab.
Under the **Secure Communications** section, click **Server Certificate**.
This will start the Web Site Certificate Wizard. Click **Next**.
Choose the **Assign an existing certificate** option and click **Next**.
You will now see a screen showing that contents of your computer's personal certificate store. Highlight your Web server certificate (denoted by the common name), and then click **Next**.
You will now see a summary screen showing you all the details about the certificate you are installing. Be sure that this information is correct or you may have problems using SSL or TLS in HTTP communications. Click **Next**, and then click **OK** to exit the wizard.
You should now have an SSL/TLS-enabled Web server. Be sure to protect your PFX files from any unwanted personnel.

To enable client authentication:

- In the MMC Console Right-click the Certificates(Local Computer)\Trusted Root Certification Authorities\Certificates folder.
- Click "All Tasks\Import".
- Follow the wizard to install ca.crt

Open the Internet Services Manager (under Administrative Tools) and navigate to the Web site you want to enable client authentication on.
Right-click on the site and click **Properties**.
You should now see the properties screen for the Web site. Click the **Directory Security** tab.
Under the **Secure Communications** section, click **Edit**.
UnFlag Require secure channel (ssl communication possible but not required).
Select Client certificates->Accept client certificates (allow users with client certificate access).
Flag Enable certificate trust list.
Press **New** and **Add from store**
Select the certificate (RTX).
Give it a friendly name (GatewayCA). -> **Finish**.

11.8 Configuring the RTX337x for server connection

This section lists the commands which are used in a basic configuration including secure data transmission using SSL.

NOTE: For the RTX337x to comply with HIPAA, SSL must be enabled.

11.8.1 Configuring the Internet connection

In order for data to be transmitted to the web server some additional commands have to be used to configure the RTX337x.

Command	Description	Details
AT+pISP	Sets or requests the phone number to the ISP (Internet Service Providers) dial-in service.	Refer to section 9.2.1, AT+pISP.
AT+pUSR	Sets or request the user name used for login to the ISP (Internet Service Provider).	Refer to section 9.2.29, AT+pUSR.
AT+pPSW	Sets or request the password used for login to the ISP (Internet Service Provider).	Refer to section 9.2.31, AT+pPSW.
AT+pWSPATH	Set or request Web Server Path	Refer to section 9.2.28, AT+pWSPATH.
AT+pWS	Sets the address of the web server the RTX337x connects to in order to transmit the collected data.	Refer to section 9.2.27, AT+pWS.
AT+pDNS	Sets the numeric IP address of the DNS servers to be used.	Refer to section 9.2.33, AT+pDNS.

Depending on the telephone system the RTX337x is attached to, the modem may need some initial setup. Please see section 9.1.38, Commands for RTX337x Communication (Phone dialing), for further details.

11.8.2 Configuring the RTX337x for SSL

The RTX337x is per default configured without any Client/Server certificates.
The following commands are available for enabling the SSL protocol. Please see section 0, Security Commands, for details.

Command	Description	Details
AT+pSAUT	Sets (or clears) SSL server authentication.	Refer to section 9.3.1, AT+pSAUT.
AT+pCAUT	Sets (or clears) SSL client authentication.	Refer to section 9.3.2, AT+pCAUT.
AT+pCACRT	Sets or request the CA (Certification Authority) certificate.	Refer to section 9.3.3, AT+pCACRT.
AT+pCCRT	Sets or request the client certificate.	Refer to section 9.3.4, AT+pCCRT.

12 XML

Extensible Markup Language (XML) is a standardized text format specially designed for transmitting structured data to Web applications. The language addresses the needs of Web publishers who encounter limitations in the ability of HTML to express structured data.

12.1 Web Service Description Language

SOAP (RPC) is used as the interface protocol. The specific interface consists of one operation which is defined in wsdl as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="RTX33XXSoap"
  targetNamespace="http://www.rtxhealthcare.com/RTX33XX/"
  xmlns:tns="http://www.rtxhealthcare.com/RTX33XX/"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:s="http://www.w3.org/2001/XMLSchema"
 xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
 xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xop="http://www.w3.org/2004/08/xop/include"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:RTX33XX="http://www.rtxhealthcare.com/RTX33XX/">
 <wsdl:types></wsdl:types>

 <wsdl:message name="xferSoapIn">
   <wsdl:part name="clientXMLString" type="xsd:string"/>
   <wsdl:part name="clientData" type="xsd:base64Binary"/>
 </wsdl:message>

 <wsdl:message name="xferSoapOut">
   <wsdl:part name="serverXMLString" type="xsd:string"/>
   <wsdl:part name="serverData" type="xsd:base64Binary"/>
   <wsdl:part name="ret" type="xsd:boolean" />
 </wsdl:message>
 <wsdl:portType name="RTX33XXSoap">
   <wsdl:operation name="xfer">
     <wsdl:documentation>Service definition of function RTX33XX__xfer</wsdl:documentation>
     <wsdl:input name="RTX33XXSoapIn" message="tns:xferSoapIn"/>
     <wsdl:output name="RTX33XXSoapOut" message="tns:xferSoapOut"/>
   </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="RTX33XXSoap" type="tns:RTX33XXSoap">
   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="xfer">
     <soap:operation soapAction="http://www.rtxhealthcare.com/RTX33XX/"/>
     <wsdl:input name="RTX33XXSoapIn" >
        <soap:body use="literal" namespace="http://www.rtxhealthcare.com/RTX33XX/"
        encodingStyle="http://www.w3.org/2003/05/soap-literal"/>
     </wsdl:input>
     <wsdl:output name="RTX33XXSoapOut">
        <soap:body use="literal" namespace="http://www.rtxhealthcare.com/RTX33XX/"
        encodingStyle="http://www.w3.org/2003/05/soap-literal"/>
     </wsdl:output>
   </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="RTX33XXSoap">
    <wsdl:port name="RTX33XXSoap" binding="tns:RTX33XXSoap">
      <soap:address location="http://www.rtxhealthcare.com/RTX33XX/RTX33XX.asmx"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

The RPC has an XML (text string) part and a binary part.

In the above wsdl:

- clientXMLString: The XML generated and sent from the RTX337x to the server.
- clientData: The binary data generated and sent from the RTX337x to the server.
- serverXMLString: The XML generated and sent from the server to the RTX337x.
- serverData: The binary data generated and sent from the server to the RTX337x.
- ret: The return value from the server. ret=true is OK, and ret=false is error.

12.2 XML part

Protocol RTX-H#1

There are two possible encodings of XML telegrams “iso-8859-1” (default) and “utf-8” (used when TrueType fonts are installed). The server telegrams should answer in the same encoding as the client used.

The XML telegram from client to server looks like this:

```
<?xml version=”1.0” encoding="iso-8859-1”?>
<xml xmlns=”RTX-H#1-Client”>
    <Gateway>
        <GatewayId>……….</GatewayId>
        <Time>……….</Time>
        <MessageNumber>……….</MessageNumber>
        <DvType>……….</DvType>
        <GwInfo>……….</GwInfo>
        <GwStatus>……….</GwStatus>
        <GwChksum>……….</GwChksum>
    </Gateway>
    <Monitor>
        <DeviceId>……….</DeviceId>
        <Type>……….</Type>
        <UtcTime>……….</UtcTime>
        <BatteryStatus>……….</BatteryStatus>
        <TimeDiff>……….</TimeDiff>
        <Value>……….</Value>
       <DvChksum>……….</DvChksum>
    </Monitor>
</xml>
```

The formal declaration of the client XML is described in RTX-H#1-Client.xsd schema:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema elementFormDefault="qualified" xmlns="RTX-H#1-Client"
xmlns:xs=http://www.w3.org/2001/XMLSchema targetNameSpace=”RTX-H#1-Client”>
     <xs:annotation>
          <xs:documentation>
               Client XML formal declaration.
          </xs:documentation>
     </xs:annotation>
     <xs:element name="xml" type="ClientXMLRecord" />
     <xs:complexType name="ClientXMLRecord">
          <xs:all>
               <xs:element name="Gateway" type="GatewayRecord" />
               <xs:element name="Monitor" type="MonitorRecord" />
          </xs:all>
     </xs:complexType>
     <xs:complexType name="GatewayRecord">
          <xs:all>
               <xs:element name="GatewayId" type="xs:hexBinary" />
               <xs:element name="Time" type="xs:int" minOccurs="0" maxOccurs="1" />
               <xs:element name="MessageNumber" type="xs:unsignedInt" />
               <xs:element name="DvType" type="xs:string" minOccurs="0" maxOccurs="1" />
```

```
            <xs:element name="GwInfo" type="xs:string" />
            <xs:element name="GwStatus" type="xs:unsignedInt" minOccurs="0" maxOccurs="1" />
            <xs:element name="GwChksum" type="xs:unsignedInt" />
        </xs:all>
    </xs:complexType>
    <xs:complexType name="MonitorRecord">
        <xs:all>
            <xs:element name="DeviceId" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="Type" type="TypeString" />
            <xs:element name="UtcTime" type="xs:int" minOccurs="0" maxOccurs="1" />
            <xs:element name="BatteryStatus" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="TimeDiff" type="xs:int" minOccurs="0" maxOccurs="1" />
            <xs:element name="Value" type="ValueString" minOccurs="0" maxOccurs="1" />
            <xs:element name="DvChksum" type="xs:unsignedInt" />
        </xs:all>
    </xs:complexType>
    <xs:complexType name="ValueString" mixed="true">
        <xs:all>
            <xs:element name="SubDev" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="SubValue" type="xs:string" minOccurs="0" maxOccurs="1" />
        </xs:all>
    </xs:complexType>
    <xs:simpleType name="TypeString">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NORMAL" />
            <xs:enumeration value="REQUESTREGISTRATION" />
            <xs:enumeration value="REQUESTFACTORYTEST" />
            <xs:enumeration value="REQUESTPATIENTINITIALIZATION" />
            <xs:enumeration value="TEST" />
            <xs:enumeration value="NOTIFICATION" />
            <xs:enumeration value="REQUEST" />
            <xs:enumeration value="RESPONSE" />
            <xs:enumeration value="USERRESPONSE" />
            <xs:enumeration value="LOG" />
            <xs:enumeration value="SENDRFU" />
            <xs:enumeration value="HEREIAM" />
            <xs:enumeration value="FORCECONNECT" />
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

The XML telegram from the server looks like this;

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<xml xmlns=”RTX-H#1-Server”>
    <Gateway>
        <GatewayId>……….</GatewayId>
        <Time>……….</Time>
        <GwChksum>……….</GwChksum>
    </Gateway>
    <Monitor>
        <Type>……….</Type>
        <Value>……….</Value>
        <DvChksum>……….</DvChksum>
    </Monitor>
</xml>
```

The formal declaration of the server XML is described in RTX-H#1-Server.xsd schema:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema elementFormDefault="qualified" xmlns="RTX-H#1-Server"
xmlns:xs=http://www.w3.org/2001/XMLSchema targetNameSpace=”RTX-H#1-Server”>
    <xs:annotation>
        <xs:documentation>
```

```
            Server XML formal declaration.

        </xs:documentation>
    </xs:annotation>
    <xs:element name="xml" type="ServerXMLRecord" />
    <xs:complexType name="ServerXMLRecord">
        <xs:all>
            <xs:element name="Gateway" type="GatewayRecord" />
            <xs:element name="Monitor" type="MonitorRecord" />
        </xs:all>
    </xs:complexType>
    <xs:complexType name="GatewayRecord">
        <xs:all>
            <xs:element name="GatewayId" type="xs:hexBinary" />
            <xs:element name="Time" type="xs:int" />
            <xs:element name="GwChksum" type="xs:unsignedInt" />
        </xs:all>
    </xs:complexType>
    <xs:complexType name="MonitorRecord">
        <xs:all>
            <xs:element name="Type" type="TypeString" />
            <xs:element name="Value" type="xs:string" minOccurs="0" maxOccurs="1" />
            <xs:element name="DvChksum" type="xs:unsignedInt" />
        </xs:all>
    </xs:complexType>
    <xs:simpleType name="TypeString">
        <xs:restriction base="xs:string">
            <xs:enumeration value="SENDLOG" />
            <xs:enumeration value="CONFIG" />
            <xs:enumeration value="NONE" />
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

NOTE: The order of the used XML tags is not fixed. The application must identify the single XLM tags and not expect the same order from one data sampling to another.

Tag	Value	Description	Xmlns
From client to server the <Type> values can be:	NORMAL	This is for an ordinary data telegram with <GatewayId> identifying the gateway and <DeviceId> identifying the External Device. Value is the value string for the External Device.	RTX-H#1-Client
	REQUESTREGISTRATION	This is a request from the client to be registered on the server.	RTX-H#1-Client
	REQUESTFACTORYTEST	This is a request from the client to be initialized with configuration data for factory testing.	RTX-H#1-Client
	REQUESTPATIENTINITIALIZATION	This is a request from the client to be initialized with configuration data for normal use.	RTX-H#1-Client
	TEST	This is a test data telegram, but otherwise with parameters as for NORMAL.	RTX-H#1-Client

	NOTIFICATION	This is a notification telegram. The Value indicates the cause of notification. The other parameters identify the source of the notification.	RTX-H#1-Client
	REQUEST	This is a request from the client to the server for actions to be performed by the client. Only the <GatewayId> has significance. The response from the server is a SENDLOG or CONFIG or NONE telegram.	RTX-H#1-Client
	RESPONSE	The <Value> parameter holds the results of a CONFIG telegram from the server.	RTX-H#1-Client
	USERRESPONSE	The <Value> parameter holds collected user responses from question tree handling.	RTX-H#1-Client
	LOG.	This is a log telegram. The log is in the <Value> parameter. Sent as a response to a SENDLOG telegram from the server.	RTX-H#1-Client
	SENDRFU.	This is a telegram telling the server, that the client wants a remote firmware download. On the next REQUEST from the client, the server should begin a firmware upload sequence through one or more CONFIG telegrams.	RTX-H#1-Client
	HEREIAM.	This is a telegram telling the server that a new External Device has been connected to the RTX337x. This gives the server a chance to configure the new External Device. The RTX337x also transmits this telegram when it boots.	RTX-H#1-Client
	FORCECONNECT	This telegram forces a connection to the server.	RTX-H#1-Client
From server to client the <Type> values can be:	SENDLOG	This is a request for the client to send the log in a LOG telegram.	RTX-H#1-Server
	CONFIG	This is new configuration information for the client. In the format of AT commands in the <Value> parameter, if any binary data is referenced in the commands this should either be included as one of the records in “serverData” or have been in “serverData” in a previous telegram (uploaded data is cleared on AT+pUNLCK).	RTX-H#1-Server
	NONE	This is a request for the client to stop asking about more requests.	RTX-H#1-Server

<GatewayId>:		Identifies the RTX337x.	RTX-H#1-Client / RTX-H1#1-Server
<DeviceId>:		Unique identification of the External Device or “Gateway” if gateway generated data. Not used in xml telegram from server to client. For devices, which requires secondary telegrams (USERRESPONSE types) to be generated, the device id consists of the unique identification concatenated with : the content of the <MessageNumber> tag. This identification is then also used for an accompanying USERRESPONSE type telegram.	RTX-H#1-Client
<DvType>:		Name of device object used to communicate with External Device. Not used in xml telegram from server to client.	RTX-H#1-Client
<Time>:		Time as seconds since 1st January 1970, for the client this represents the time the telegram was created. For the server this is the current time for the RTX337x decided by the server.	RTX-H#1-Client / RTX-H1#1-Server
<Value>:		Is the value string. Its content depends on the <Type> and on the device. Inside the value tag there can be other tags depending on the device. See section 12.2.1, Value Formats and chapter 13, Installation of External Devices, for details.	RTX-H#1-Client / RTX-H1#1-Server
<GwChksum>:		Adler-32 based checksum of the data inside the <GatewayId>, <Time>, <MessageNumber>, <DvType>, <GwInfo> and the <GwStatus> added to the list tags. The tags themselves are NOT included. The checksum is an ASCII representation of a 32 bit unsigned value (decimal) and startvalue (seed) is 1.	RTX-H#1-Client / RTX-H1#1-Server
<DvChksum>:		Adler-32 based checksum of the data inside the <DeviceId>, <Type>, <UtcTime>, <BatteryStatus>, <TimeDiff> and <Value> tags. The tags themselves are NOT included. Subtags inside the <value> tag are not included in the calculation either. The checksum is an ASCII representation of a 32 bit unsigned	RTX-H#1-Client / RTX-H1#1-Server

		value (decimal) and startvalue (seed) is 1.	
<Gateway>:		RTX337x specific data. Data in Subtags all relates to data from/to the RTX337x.	RTX-H#1-Client / RTX-H1#1-Server
<Monitor>:		Monitor specific data. The External Device or the RTX337x, depending of <Type> generates the data.	RTX-H#1-Client / RTX-H1#1-Server
<UtcTime>:		The UtcTime is the time stamped by the External Device when the measurement has been taken. Not used in xml telegram from server to client.	RTX-H#1-Client
<MessageNumber>		Number of the message. This number is automatically incremented by the RTX337x every time a message is generated. Not used in xml telegram from server to client.	RTX-H#1-Client
<BatteryStatus>		This tag is used to send battery status to the server, if the RTX337x has this information from the External Device. Not used in xml telegram from server to client.	RTX-H#1-Client
<TimeDiff>		This tag is used to send the difference in seconds between the external device and RTX337x (deviceTime – RTX337xTime) real time clocks for devices which do not support setting their clocks. Not used in xml telegram from server to client.	RTX-H#1-Client
<GwInfo>		Order code, Device master record and the software version of the RX337x. Not used in xml telegram from server to client.	RTX-H#1-Client
<GwStatus>		Status of the RTX337x. Not used in xml telegram from server to client.	RTX-H#1-Client

12.2.1 Value formats

When the <Value> tag holds data from an external device, the general format is: <Value><SubDev>(valueformat)</SubDev><SubValue>(data)</SubValue></Value>, where the data format is defined in the <SubDev> tag and the data in the <SubValue> tag.

Example:

```
<Value>
        <SubDev>Bpm1</SubDev>
        <SubValue>Sys:128 mmHg Dia:083 mmHg Pulse:066 1/min MAP:000 mmHg</SubValue>
</Value>
```

At present these value formats are defined:

12.2.1.1 Blood Glucose Meter

Bgm1

Format: aaaaa_bbbbbb_cccc

aaaaa Measurement value, might include a dot as separator, left padded with zero. Might include special values “HI   ‟ and “LO   ‟.

bbbbbb Unit. If unit has less than six characters it is right padded with spaces

cccc Status flag, 4 digit ASCII hex string representation of status flags. Details can be found in the table below.

Bit 0	Temperature out of range at measurement time.
Bit 1	Measurement marked as control level 1.
Bit 2	LO. Measurement below low setting on monitor.
Bit 3	HI. Measurement above high setting on monitor.
Bit 4	HYPO. Measurement below hypo setting on monitor.
Bit 5	Data/time not set
Bit 6	Measurement marked as control level 2
Bit 7	Used drum/barcode
Bit 8	Expired drum/strip (for Roche Aviva/Performa this indicates strips expire in 30 days).
Bit 9	General Flag, Asterisk
Bit 10	Result over personal target or control result above controls range
Bit 11	Result below personal target or control result below controls range
Bit 12	Control not identified
Bit 13-15	Unused

_ Space character.

Examples:	00107 mg/dL 0000	Ok.
	005.8 mmol/L 0000	OK.
	00104 mg/dL 0020	Invalid time.
	HI mg/dL 0008	Measurement above high setting on monitor.
	LO mg/dL 0004	Measurement below low setting on monitor.
	00101 mg/dL 0002	Measurement marked as control.

Bgm2

Format: aaaaa_bbbbbb_cccc_d_ee

aaaaa Measurement value, might include a dot as separator, left padded with zero. Might include special values “HI   ‟ and “LO   ‟.

bbbbbb Unit. If unit has less than six characters it is right padded with spaces

cccc Status flag, 4 digit ASCII hex string representation of status flags. Details can be found in the table below.

d Meal flag ,1 ASCII character representation of Meal flags. Details can be found in the table below.

ee Meal comment, 2 digit ASCII decimal string representation of meal comment. Details can be found in the table below.

Bit 0	Temperature out of range at measurement time.
Bit 1	Measurement marked as control level 1.
Bit 2	LO. Measurement below low setting on monitor.
Bit 3	HI. Measurement above high setting on monitor.
Bit 4	User-marked control
Bit 5	User-deleted result
Bit 6	Self detected control
Bit 7-12	Unused
Bit 13	Parity error
Bit 14-15	Unused

Description of Meal flag values.

N	No Flag allocated.
B	User flags the record as being taken before a meal.
A	User flags the record as being taken after a meal.
F	User flags the record as being taken fasting.
D	Logbook

Description of Meal comment definitions.

00	No Comment.
01	Not Enough Food
02	Too Much Food
03	Mild Exercise
04	Hard Exercise
05	Medication
06	Stress
07	Illness
08	Feel Hypo
09	Menses
10	Vacation
11	Other

Examples:

Configured to mg/dL and everything ok

```
<Value>
        <SubDev>Bgm2</SubDev>
        <SubValue>00107 mg/dL  0000 N 00</SubValue>
</Value>
```

Configured to mmol/L and a control measurement

```
<Value>
        <SubDev>Bgm2</SubDev>
        <SubValue>005.8 mmol/L 0002 N 00</SubValue>
</Value>
```

Configured to mg/dL and User Meal Flags "After Meal" and "Too Much Food"

```
<Value>
        <SubDev>Bgm2</SubDev>
        <SubValue>00117 mg/dL  0000 A 02</SubValue>
</Value>
```

Bgm3

Format: aa_bbbbbb
aa Record type. 2 digit ASCII decimal string representation of type number.
bbbbbb Record content, 6 digit ASCII hex string representation of Record content. Details can be found in the table below.

Description of Record Content.
Glucose Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
0	Glucose	Value: 0-1023 Commenting (see list of comments below)	1	10 14	Meter always stores mg/dl.

Description of Comments.

Bit	Type		Value
Bit 10 (LSB)	Control solution	Blood (normal or alternate site),	0
		Control	1
Bit 11	Alternate site	Normal test (blood taken from finger tips)	0
		Alternate site test	1
Bit 12-14	Meal	No food commenting	0
		Before breakfast	1
		after breakfast	2
		Before lunch	3
		after lunch	4
		Before dinner	5
		after dinner	6
		Night	7
Bit 15	Reserved		-
Bit 16-17	Exercise	No exercise comment	0
		Before exercise	1
		During exercise	2
		After exercise	3
Bit 18	Health	Stress	0/1
Bit 19		Feel hypo	0/1
Bit 20		Illness	0/1
Bit 21		Menses	0/1
Bit 22		Vacation	0/1
Bit 23		Other	0/1

Food Records:

Number	**Description**	**Range**	**Resolution**	**Used bits in Record data (LSB first)**	**Comment**
20	Meal 1	Meal segment: brkf 0 lunch 1 dinner 2 snack 3 alc 4	1	3	Carbs and fats are invalid if meal segment alcohol is set.
		Carbs: 0 . 250: 0 - 250 "---": 255	1	8	
		Fats: 0 . 250: 0 - 250 "---": 255	1	8 5 left	

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
21	Meal 2 Protein	Meal segment: brkf 0 lunch 1 dinner 2 snack 3 alc 4 Cal: 0 - 2500: 0 - 500 "---": 511 Prot: 0 - 250: 0 - 250 "---": 255	1 5 1	3 9 8 4 left	Calories and proteins are invalid if meal segment alc is set.

Health Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
30	Ketones	neg. 0 trace 1 small 2 moderate 3 large 4	1	3 21 left	
31	HbA1C	4 - 15 % 40 -150	0.1 %	8 16 left	
32	Micro albumin	normal 0 positive 1 1 2 to 1000 1001 >1000 1002	1	10 14 left	
33	Cholesterol LDL, HDL	LDL: 0 - 500 0 - 500 >500 501 "---" 511 HDL: 0 - 500 0 - 500 >500 501 "---" 511	1 mg/dl 1 mg/dl	9 9 6 left	

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
34	Cholesterol Total, TG (Triglyceride)	Total: 0 - 1000 0-1000 >1000 1001 "---" 1023 TG: 0 - 3000 0-3000 >3000 3001 "---" 4095	1 mg/dl 1 mg/dl	10 12 2 left	Meter stores values in mg/dl but always displays in selected units Conversion between mg/dl and mmol/l for Total: Factor = 38,61 [6] Range 0 - >25.9 mmol/l 1001 mg/dl <> >25.9 mmol/l Conversion between mg/dl and mmol/l for TG: Factor = 88,5 [6] Range 0 – >33.9 mmol/l 3001 mg/dl <> >33.9 mmol/l
35	Blood pressure	80 - 200 80 - 200 40 - 150 40 - 150	1	9 9 6 left	
36	Weight height	Weight: 0 - 600 0 - 1200 "---" 2047 Height: 0 - 255 0 - 510 "---" 511	0.5 0.5	11 9 4 left	
37	Health notes	Stress Feel hypo Illness Menses Vacation Other	-	1 1 1 1 1 1 18 left	
38	Last visit	Doctor 0 Eye exam. 1 Foot exam. 2	1	2 22 left	

Medication Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
10	Oral medication pills intake	Pill type: 1 - 15 1 – 15 Number: 0 - 5 0 - 10	1 0.5	4 4 16 left	Type 0: None 5 names fixed (from language table), 10 names customizable via One Touch DMS.

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
11	Insulin injection	Insulin type: 1 - 21 1 – 21 Units: 0 - 250 0 - 2500	1 0.1	5 12 7 left	Type 0: None 10 names fixed (from language table), 11 names customizable via One Touch DMS.
12	Setting Bolus	Units: 0 - 25 0 -250	0.1	8 16 left	
13	Setting Pump Daily Total	Units: 0 - 100 0 -1000	0.1	10 14 left	

Exercise Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
45	Exercise	Level: mild 0 moderate 1 hard 2 Duration: 0 - 24h 0 - 288	1 5 min.	2 9 13 left	

Examples.

Measurement 379 mg/dl with flag “Before lunch”

```
<Value>
          <SubDev> Bgm3</SubDev>
          <SubValue> 00 00317B</SubValue>
</Value>
```

Type 1 Insulin 20.3 UI.

```
<Value>
          <SubDev>Bgm3</SubDev>
          <SubValue> 11 001961</SubValue>
</Value>
```

12.2.1.2 Blood Pressure Monitor

Bpm1

Format: Sys:aaa_bbbb_Dia:ccc_bbbb_Pulse:ddd_eeeee_MAP:fff_bbbb

aaa Systolic pressure. If value has less than three characters, it is left padded with zeros.
bbbb Pressure unit. If unit has less than four characters it is right padded with spaces.
ccc Diastolic pressure. If value has less than three characters, it is left padded with zeros.
ddd Pulse (heart rate). If value has less than three characters, it is left padded with zeros.
eeeee Pulse unit. If unit has less than five characters it is right padded with spaces.
fff Mean arterial pressure. If value has less than three characters, it is left padded with zeros. If MAP is not measured by the monitor, the value is 000.
_ Space character.

Examples: Sys:156 mmHg Dia:090 mmHg Pulse:097 1/min MAP:103 mmHg
Sys:109 mmHg Dia:081 mmHg Pulse:068 1/min MAP:000 mmHg

12.2.1.3 Weight Scale

Ws1

Format: abbbbbb_cc

a Sign (- or +)
bbbbbb Weight. The value might include decimal dot. If value has less than six characters, it is left padded with zeros.
cc Unit. If unit has less than two characters it is right padded with spaces.
_ Space character.

Examples: +0185.3 lb
-0001.5 lb
+071.35 kg

Example of a full XML telegram, containing data from a A&D weight scale:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B0063B6</GatewayId>
    <GwInfo>50100002, XX, RTX337x-1_3</GwInfo>
    <DvType>ADBTWSType</DvType>
    <GwChksum>2743340561</GwChksum>
    <Time>1182252890</Time>
    <MessageNumber>51</MessageNumber>
  </Gateway>
  <Monitor>
    <BatteryStatus>206</BatteryStatus>
    <Value>
      <SubDev>Ws1</SubDev>
      <SubValue>+084.60 kg</SubValue>
    </Value>
    <UtcTime>1182252888</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>4286253795</DvChksum>
    <DeviceId>00A0960D6981:51</DeviceId>
  </Monitor>
</xml>
```

12.2.1.4 Spirometer

Spiro-1

Format: FVC:_aaaa_ml,_PEF:_bbb_l/min,_FEV1:_cccc_ml,_FEF75:_
dddd_ml/s,_FEF50:_eeee_ml/s,_FEF25_:ffff_ml/s,_FEF75_25_:gggg_ml/s,_Mode:h

aaaa FVC Measurement value in ml (value might be -1 if that particular measurement value was unavailable)

bbb PEF Measurement value in l/min (value might be -1 if that particular measurement value was unavailable)

cccc FEV1 Measurement value in ml (value might be -1 if that particular measurement value was unavailable)

dddd FEF75 Measurement value in ml/s (value might be -1 if that particular measurement value was unavailable)

eeee FEF50 Measurement value in ml/s (value might be -1 if that particular measurement value was unavailable)

ffff FEF25 Measurement value in ml/s (value might be -1 if that particular measurement value was unavailable)

gggg FEF75_25 Measurement value in ml/s (value might be -1 if that particular measurement value was unavailable)

h Mode

- 0 = Premesurement
- 1 = measurement medication
- 2 = measurement events
- 3 = measurement symptoms

_ Space character.

_ _

Example:
<Value>
<SubDev>Spiro-1</SubDev>
<SubValue>FVC: 3081 ml, PEF: 160 l/min, FEV1: 2056 ml, FEF75: 0079 ml/s, FEF50: 0180 ml/s, FEF25: 0246 ml/s, FEF75_25: 0145 ml/s, Mode: 0<SubValue>
</Value>

Spiro-2

Format: FVC:_aaaa_cL, FEV1:_bbbb_cL, FEV1%:_ccc, PEF:_dddd_L/min,
FEF2575:_eeee_cL/s, FET:_ffff_1/10s, SYMPTOM:_gggggggg, QUESTION:_hhhh,
QUALITY:_ii, SW:_jjjj

aaaa FVC (Forced Vital Capacity) in cL (value might be -1 if that particular measurement value was unavailable)

bbbb FEV1 (Volume expired in the 1st second of test) in cL (value might be -1 if that particular measurement value was unavailable)

ccc FEV1% (FEV1/FVC*100) in percent (value might be -1 if that particular measurement value was unavailable)

dddd PEF (Peak Expiratory Flow) in L/min (value might be -1 if that particular measurement value was unavailable)

eeee FEF50 Measurement value in ml/s (value might be -1 if that particular measurement value was unavailable)

ffff FET (Forced Expiratory Time) in 10ths of a second (value might be -1 if that particular measurement value was unavailable)

gggggggg SYMPTOM Answers to the 8 symptom questions. One digit to each symptom, first digit to symptom1, next to symptom2 and so on.
0 Not activated
1 No
2 Average

3 Maximum

hhhh QUESTION Answers to the 4 questions. One digit to each question, first digit to question1, next to question2 and so on.

0 Not Activated
1 Yes
2 No

ii QUALITY One byte hex format report of the measurement quality:
Bit format:
0x01 No problem
0x02 Repeat test and start faster
0x04 Repeat test without coughing
0x08 Breathe out for a longer time
0x10 Breathe out all air in the lungs
0x20 Warning! Values dropping
0x40 Warning! There is variability
0x80 The spirometry is

jjjj Software version.

_ Space character.

Example:
<Value>
<SubDev>Spiro-2</SubDev>
<SubValue>FVC: 0500 cL, FEV1: 0484 cL, FEV1%: 083, PEF: 0966 L/min, FEF2575: 0451 cL/s, FET: 0444 1/10s, SYMPTOM: 03000000, QUESTION: 0100, QUALITY: 01, SW: 0304<SubValue>
</Value>
(SYMPTOM: Chest Tightness=Maximum, QUESTION: Drug taken=Yes, QUALITY: No problem)

12.2.1.5 AM1 Key

AM1Key-1

Format: Key:_abcd

Key values:

a	Indicates type (1 = Medication, 2= Event, 3= Symptoms).
b	For Medication and Event b is the value 0-3, for Symptoms b= Cough value 0-3
c	For Medication and Event 0, for symptoms c= Dyspnea value 0-3.
d	For Medication and Event 0, for symptoms d= Sputum value 0-3.

Example:
<Value>
<SubDev>AM1Key-1</SubDev>
<SubValue>Key: 1200<SubValue>
</Value>

12.2.1.6 Oximeter (SpO$_2$)

SpO2-1

Format: HR:aaa bpm_SpO2:bbb %_cccc

aaa	Heart Rate.
bbb	SpO2 value
cccc	Status flag, 4 digit ASCII hex string representation of status flags. Details can be found in the table below.
_	Space character

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Green perfusion
Bit 1	Yellow perfusion
Bit 2	Red perfusion
Bit 3	Artifact
Bit 4-15	Unused

Example:

```
<Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:072 bpm SpO2:097 % 0001</SubValue>
</Value>
```

12.2.1.7 Coagulation monitor

Coagu1

Format: INR:aaa.a, %Quick:bbbb, Sec:ccc.c, Unit:d, INR_L:eee.e, INR_H::fff.f, MeasNr:hhhhh, Lot:iiiii, Strip:jj, Exp:kkkkkk, Flag:llll, DateFmt:m, TimeFmt:n, Beeper:o, LimitsOn:p SWVer:qq.qq, HWVer:rrrrrrrrrrrrrr, Model:ss, MaxTrans:tt, MaxMeas:uuu, Protocol:v.v, MaxBytes:www, BootL:xx.xx, Bcycle:zzz, PCBID:IIIIIIIIIIIII

aaa.a Measurement value (INR), left padded with zeroes.
bbbb Measurement value (%Quick), left padded with zeroes.
ccc.c Measurement value (sec), left padded with zeroes.
d Unit Shown in display 1 = INR, 2 = %Quick, 3 = Sec.
eee.e INR Low setting in meter, left padded with zeroes.
fff.f INR High setting in meter, left padded with zeroes.
hhhhh Measurement sequence number, left padded with zeroes.
iiiii Strip lot number from Code Chip.
jj Type of test strip 01 = PT.
kkkkkk Strip expiry date from Code Chip in 'yymmdd' format.
llll Status flag, 4 digit ASCII hex string representation of status flags. Details can be found in the table below.
m Date format in display 1 = dd-mm-yy, 2 = mm-dd-yy and 3 = yy-mm-dd.
n Time Format in display 1 = 24 hours and 2 = 12 hours.
o Beeper 0 = off and 1 = on.
p Patient limits 0 = off and 1 = on.
qq.qq Software version.
rrrrrrrrrrrrrr Hardware version (might be up to 14 chars).
ss Model ID (might be up to 14 chars).
tt Maximum transmission time.
uuu Maximum number of measurement records.
v.v Communication protocol version
www Maximum number of bytes of a receiving data packet.
xx.xx Boot Loader version.
zzz Duration of basis cycle per 10e-5 second.

IIIIIIIIIIIII Circuit board Identification, format is: AAAYMIIXXXXXXP where:
AAA = Board number.
Y = Production year.

M = Manufacturer characteristic.
II = Index.
XXXXXX = Sequence number.
P = Checksum.

Status flag:

Bit 0	Measurement transmitted (NOTE: this does not indicate it has been transferred to the RTX337x, just that is has been transferred somewhere).
Bit 1	Date and time set by user
Bit 2	Control Measurement
Bit 3	Range Active – check of patient high and low target activated.
Bit 4	Result of sec value is lower than measurement range
Bit 5	Result of sec value is higher than measurement range
Bit 6	Result of INR value is lower than measurement range
Bit 7	Result of INR value is higher than measurement range
Bit 8	Result of %Quick value is lower than measurement range
Bit 9	Result of %Quick value is higher than measurement range
Bit 10-15	Unused

Example:

```
<Value>
       <SubDev>Coagu1</SubDev>
        <SubValue>INR:000.9, %Quick:0111, Sec:010.7, Unit:1, INR_L:001.5,
INR_H:002.5, MeasNr:00001, Lot:00005, Strip:15, Exp:051200, Flag:0005, DateFmt:1,
TimeFmt:1, Beeper:1, LimitsOn:1 SWVer:03.02, HWVer:00000000000000, Model:UP,
MaxTrans:12, MaxMeas:100, Protocol:1.1, MaxBytes:180, BootL:00.58, Bcycle:330,
PCBID:4104103001979P</SubValue>
</Value>
```

12.2.1.8 ECG recorder

Ecg1

Format: aaa_bbbbbbbbbb

aaa	Header size in bytes. If value has less than three characters, it is left padded with zeros.
bbbbbbbbbb	ECG data size in bytes.
_	Space character.

Example: 123 1234567890

12.3 Binary part

The binary data consists of a number of records. Each record has the format:

The first field is the length of the content field in bytes. Size – 4bytes (Little Endian format)
The next field is the name (in letters and/or digits) as a string terminated with binary zero. This field identifies the record and must be unique. Size – 11 bytes, right padded with binary zero(s).
The last field of the record is the content. Size – length bytes

The last record has a length field, which is zero and no name field. If there are no binary data this record must still be present.

The content of the content field is the binary data. There are no restrictions on the content.

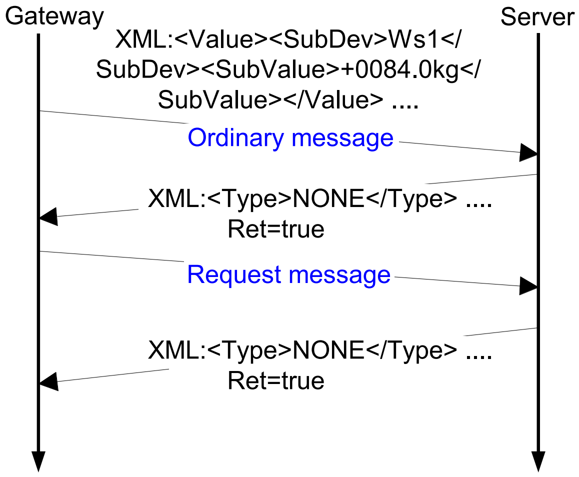
Integrity of the binary data is not checked at the application layer but is left up to TCP and if used SSL. An exception to this is the firmware upload package, which will be verified before the update is performed.

12.4 Server communication sequence

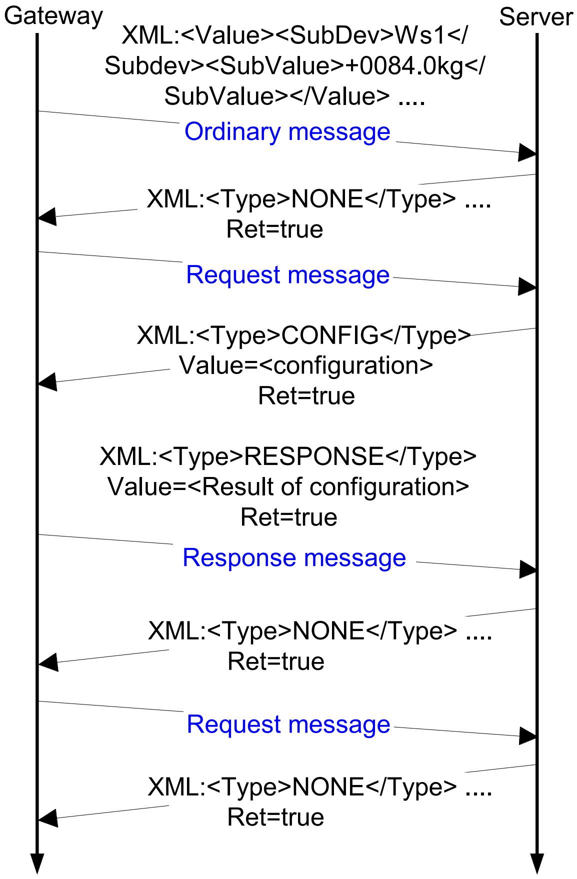
The sequence of events is a follows:

1. Connect to the server via the Internet and the ISP.
2. If the queue for telegrams to the Internet server is empty then stop.
3. Send data to the server packed as an XML string. The <type> tag identifies the kind of data in the XML string. The type can be one of the following: NORMAL, REQUESTREGISTRATION, REQUESTFACTORYTEST, REQUESTPATIENTINITIALIZATION, TEST, NOTIFICATION, SENDRFU, HEREIAM, FORCECONNECT, USERRESPONSE.
4. Ask server if there are any requests (<type> tag of XML string is REQUEST) or send the response to the last command (<type> RESPONSE or LOG).
5. Receive data from the server. Data format is the same as above. If <type> tag is NONE and the last transmitted telegram type was REQUEST then continue with 8, otherwise continue with 4.
6. If there are requests then process the requests. The type can be one of the following SENDLOG, CONFIG.
7. Continue with 4.
8. If client server turnaround time is greater than 20 sec. then continue with 2.
9. If client time deviates more than 5 sec. from the returned xml telegram server time (in the <time> tag) then synchronize client time with server time.
10. Continue with 2.

Example with 1 packet of measuring weight data (XML: is the content of the XML telegram).


Gateway
Server
XML:<Value><SubDev>Ws1</SubDev><SubValue>+0084.0kg</SubValue></Value> ....
Ordinary message
XML:<Type>NONE</Type> ....
Ret=true
Request message
XML:<Type>NONE</Type> ....
Ret=true

Example with 1 packet of measuring Weight data and new configuration from the server


Gateway
Server
XML:<Value><SubDev>Ws1</Subdev><SubValue>+0084.0kg</SubValue></Value> ....
Ordinary message
XML:<Type>NONE</Type> ....
Ret=true
Request message
XML:<Type>CONFIG</Type>
Value=<configuration>
Ret=true
XML:<Type>RESPONSE</Type>
Value=<Result of configuration>
Ret=true
Response message
XML:<Type>NONE</Type> ....
Ret=true
Request message
XML:<Type>NONE</Type> ....
Ret=true

13 Installation of External devices

The RTX337x TeleHealth Monitor is used for reception and communication of measurements taken by on-site external devices such as Blood Pressure Monitors, Personal Weight Scales, Blood Glucose Meters and other compatible devices. The communication will be performed using:

- Wireless Bluetooth data connection.
- Infrared data connection.
- Cable (RS232) connection.

External devices to be used with RTX337x:

External Device	Interface to RTX337x	DeviceType
A&D Bluetooth Weight Scale "UC-321PBT" & "AD-6121ABT1"	Wireless Bluetooth connection	**ADBTWSType**
IEM Bluetooth Weight Scale (TC 100 and Libr-O-Graph)	Wireless Bluetooth connection	**IEMBTWSType**
A&D Serial Weight Scale "UC-321PL"	Cable (RS232) connection	**ADSerScaleType**
THT Bluetooth Weight Scales "SC-1 and SC-2"	Wireless Bluetooth connection	**THTBTWSType**
THT Serial Weight Scale "SC-1"	Cable (RS232) connection	**THTSerScaleType**
A&D Bluetooth Blood Pressure Monitor "UA-767PBT"	Wireless Bluetooth connection	**ADBTBPType**
IEM Bluetooth Blood Pressure Monitor (Stabil-O-Graph)	Wireless Bluetooth connection	**IEMBTBPType**
A&D Serial Blood Pressure Monitor "UA-767PC"	Cable (RS232) connection	**ADSerBPMType**
Roche "Accu-Chek Compact", "Accu-Chek Compact Plus", "Accu-Chek Aviva", "Accu-Chek Active", "Accu-Chek Performa", "Accu-Chek Aviva Nano" Blood Glucose meters and Roche "CoaguCheck XS" coagulation monitor.	Infrared connection	**GenIR1Type** **GenIR1TSType**
Lifescan "OneTouch Ultra" Blood Glucose meter	Cable (RS232) connection	**LsBgmType**
Lifescan "OneTouch Ultra 2" Blood Glucose meter	Cable (RS232) connection	**LsBgm2Type**
LifeScan "OneTouch UltraEasy" Blood Glucose meter	Cable (RS232) connection	**LsEasyType**
LifeScan "OneTouch Vita" Blood Glucose meter	Cable (RS232) connection	**LsVitaType**
LifeScan "OneTouch UltraSmart" Blood Glucose meter	Cable (RS232) connection	**LsSmartType**
Abbot/Therasense "FreeStyle", "FreeStyle Flash", "FreeStyle Lite" and "FreeStyle Freedom Lite" Blood Glucose meters	Cable (RS232) connection	**FreeStyleType**
Bayer "BREEZE®", "BREEZE2®" and "CONTOUR®" (15- and 5-seconds version)	Cable (RS232) connection	**BayerSerBgmType**
ViaSys AM1(+) Asthma monitor	Wireless Bluetooth connection	**ViasysAM1BTType**

MIR Spirotel Spirometer	Cable (RS232) connection	**MirSpirotelType**
Nonin Ipod Oximeter	Cable (RS232) connection	**NoninIpodType**
Nonin 4100BT Oximeter	Wireless Bluetooth connection	**Nonin4100BTType**
Nonin 9560 OnyxII BT Oximeter	Wireless Bluetooth connection	**NoninOnyx2BTType**
Mindray PM50 Oximeter	Cable (RS232) connection	**MindrayPM50Type**
Mindray PM60 Oximeter	Infrared connection	**MindrayPM60Type**
Corscience 3/6 and 12 BT ECG recorders	Wireless Bluetooth connection	**CS3612ECGType**

For more specific descriptions of compatible devices and how to configure for use with the RTX337x read the following sections.
A virtual device (internal device in the RTX337x) is also described in this chapter. The virtual device covers the functionality of reminders and server connection supervision.

13.1 External Device Driver Enabling

Most external devices are per default enabled, though in rare cases some devices require an activation code for enabling. If this is the case, it will be explicitly stated in the description of the device.

When inserting an external device using the AT+pINSDV command the user will be advised if the insertion was successful or not. If the external device type is miss-spelled the operator will receive a '**Device DEVICETYPE not permitted**' error and if the external device is not allowed (external device type is known but permission code not correct) the operator will receive a '**Device DEVICETYPE not permitted**' error message.

If an activation code is required, it must be obtained from Tunstall Healthcare A/S. Activation code is only required if it is specifically stated in the descriptions of the specific devices. Below is an example of insertion of an activation code:

```
Operator Level>AT+pCODE=30303030303030303037303030303030303030303052545820486616C7468
6361726520412F53333537323134373574831<cr>
Operator Level> OK
Operator Level> AT+pCODE?
Operator Level> Permission code: 0X0000000000003030  Customer: Tunstall Healthcare A/S
```

13.2 A&D Bluetooth weight scale

This section describes how to attach an A&D Bluetooth weight scale to the RTX337x.

13.2.1 Compatible A&D scales

- A&D Bluetooth scale UC-321PBT
- A&D Bluetooth scale AD-6121ABT1

13.2.2 Installing the A&D Bluetooth weight scale

To install an A&D weight scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The weight scale is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>*.***

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>.

<devicetype> for the A&D Bluetooth weight scale is: **ADBTWSType**.

Using Bluetooth address
Below is an example on how to insert an A&D weight scale with device name: “TestDevice” and the following Bluetooth address: 00043EC204D2. Installation using the Bluetooth address prevents unauthorized connections to take place. The Bluetooth address is placed on the backside of the weight scale as shown in Figure 15.

AT+pINSDV=TestDevice:ADBTWSType:00043EC204D2-39121440

Generic installation
The scale can also be inserted as generic, which opens the RTX337x for pairing with all A&D Bluetooth scales, no matter which Bluetooth addresses they have. The device is only open for pairing with all scales until the first successful connection. After this is completed the RTX337x will only respond to this scale and is no longer considered generic.

AT+pINSDV=TestDevice:ADBTWSType:FFFFFFFFFFFF-39121440

NOTE: To prevent conflicts it is only possible to install one type of Bluetooth product as generic at a time.

NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these

AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the folowing.
The JavaScripts will be activated with a set of parameters as described in the following.

13.2.2.1 A&D Bluetooth Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.2.2.2 A&D Bluetooth Scale <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	“39121440”
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of ‘.’ as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: “-” (0x2D)

Below is an example on how to insert the scale with device name: “TestDevice” and the following Bluetooth address: 00043EC204D2, and assigning a JavaScript named “QT1” for the Measurement Event at priority 2 and another JavaScript named “QT7” for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum measurement limit in kg.
- GeneralScriptParameter2: Maximum measurement limit in kg.
- GeneralScriptParameter3: Target weight in kg.
- GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=TestDevice:ADBTWSType:00043EC204D2-39121440-QT1,2---QT7–50.0–100.0–75-15

13.2.2.3 A&D Bluetooth Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are “doubles”)
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970 (If time is set to a date before production date this value will be -1)

2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	0-255 (Conversion formula is: VBatt = value * 0.02V + 1.9V and the battery should be replaced when VBatt < 4.6V)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.2.3 Pairing with A&D Bluetooth weight scale

To pair the weight scale with the RTX337x (which is prepared as described in section 13.2.2) some actions must be performed.

- Place the weight scale near the RTX337x (RTX337x must be powered).
- Make a measurement to initiate the data transmission.
- If the weight scale has already been paired to a RTX337x it will try to reconnect to that one.
- If the weight scale does not find the RTX337x it was already paired with, it will stop searching for 1 minute and then try again.
- If it then finds the RTX337x that it was already paired with, it delievers data.
- If it does not find the RTX337x that it was already paired with, it starts an inquiry to search for another device to pair with (any compatible Bluetooth device).
- If the weight scale finds a new compatible Bluetooth device, it pairs and delivers data.
- If the weight scale does not find a compatible Bluetooth device to pair with it stops searching and a new measurement must me made to restart the procedure.

Step by step installation

Step	Description	Tools
[1]	Configure the RTX337x. Check that the weight scale is inserted (AT+pINSDV?).	RTX337x + cable or web interface
[2]	Pair the weight scale to the RTX337x.	Weight scale and RTX337x
[3]	When a weight scale is 'paired' and a measurement is made, the RTX337x tries to connect to the server and transmit data. If transmission fails, the data will be delivered next time the RTX337x connects to the server.	Weight scale and RTX337x

13.2.4 Data to server

The <Monitor> part of the XML delivery format for the A&D Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: "Ws1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

A&D Bluetooth Scale:

Configured to kg

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>ADBTWSType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>00043EC204D2:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus>100</BatteryStatus>
   <Value>
        <SubDev>Ws1</SubDev>
        <SubValue>+084.00 kg</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

Configured to lbs

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>ADBTWSType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>00043EC204D2:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus>100</BatteryStatus>
   <Value>
        <SubDev>Ws1</SubDev>
        <SubValue>+0132.0 lb</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>………</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>00043EC204D2:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.2.5 Unique identification

The figure below shows an example of how the A&D weight scale is unique identified via a Bluetooth address. The Bluetooth address is also available as barcode.

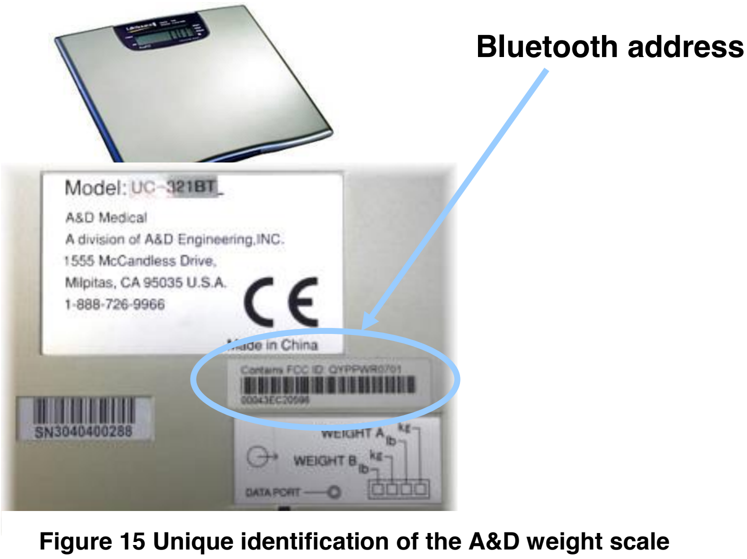


Figure 15 Unique identification of the A&D weight scale

13.2.6 Data port switch on scale

The A&D Bluetooth scale UC-321PBT has a “Data port” switch for setting the output from the scale. This switch MUST BE IN POSITION “Weight B Kg” or “Weight B Pounds”.

13.3 IEM Bluetooth weight scale

This section describes how to attach an IEM Bluetooth weight scale to the RTX337x.

13.3.1 Compatible IEM scales

- IEM Bluetooth scale (TC 100)
- IEM Bluetooth scale (Libr-O-Graph)

13.3.2 Installing the IEM Bluetooth weight scale

To install an IEM weight scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The weight scale is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>*.***

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>

<devicetype> for the IEM Bluetooth weight scale is: **IEMBTWSType**.

Using Bluetooth address
Below is an example on how to insert an IEM weight scale with device name: “TestDevice” and the following Bluetooth address: 00077290CC12. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:IEMBTWSType:00077290CC12-6624
(The Bluetooth address is not labeled on the IEM scale and if not known it may be replaced with FFFFFFFFFFFF for generic insertion.)

Generic installation
The scale can also be inserted as generic, which opens the RTX337x for pairing with all IEM Bluetooth scales, no matter which Bluetooth addresses they have. The device is only open for pairing with all scales until the first successful connection. After this is completed the RTX337x will only respond to this scale and is no longer considered generic.

AT+pINSDV=TestDevice:IEMBTWSType:FFFFFFFFFFFF-6624

NOTE: To prevent conflicts it is only possible to install one type of Bluetooth product as generic

NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of 1the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.3.2.1 IEM Bluetooth Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.3.2.2 IEM Bluetooth Scale <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"6624"
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with device name: "TestDevice" and the following Bluetooth address: 00077290CC12 and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum measurement limit in kg.
- GeneralScriptParameter2: Maximum measurement limit in kg.
- GeneralScriptParameter3: Target weight in kg.
- GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=TestDevice:IEMBTWSType:00077290CC12-6624-QT1,2---QT7-50.0–100.0–75-15

13.3.2.3 IEM Bluetooth Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value in kg.
1	Measurement Timestamp	Sec. since 1-Jan-1970
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	-1 (Not used)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.3.3 Pairing with the IEM Bluetooth scale

To pair the weight scale with the RTX337x (which is prepared as described in section 13.2.2) some actions must be performed. Place the scale near the RTX337x (RTX337x must be powered). Disconnect the battery (if connected), wait a minute and then reconnect. The scale will beep with 3 different frequencies and if successful a short beep after a few seconds.

If the pairing in not successful the scale will beep with 12 short beeps and the pairing procedure must be repeated.

Step by step installation

Step	Description	Tools
[1]	Configure the RTX337x. Check that the scale is inserted (AT+pINSDV?).	RTX337x + cable or web interface
[2]	Pair the scale to the RTX337x.	Scale and RTX337x
[3]	When a scale is 'paired' and a measurement is made, the RTX337x tries to connect to the server and transmit data. If transmission fails, the data will be delivered next time the RTX337x connects to the server.	Scale and RTX337x

13.3.4 Data to server

The <Monitor> part of the XML delivery format for the IEM Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Ws1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

IEM Scale:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>IEMBTWSType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>122001BG:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>Ws1</SubDev>
        <SubValue>+084.00 kg</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>………</Time>
      <MessageNumber>25</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>122001BG:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.4 A&D serial weight scale

This section describes how to attach an A&D serial weight scale to the RTX337x.

13.4.1 Compatible A&D scales

- A&D serial scale UC-321PL

13.4.2 Installing the A&D serial weight scale

To install an A&D serial weight scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The weight scale is installed using the **<devicetype>.**

The command for installing the serial A&D weight scale is:
AT+pINSDV=<devicename>:<devicetype>:

<devicetype> for the A&D Serial weight scale is: **ADSerScaleType**.

Below is an example on how to insert an A&D serial weight scale with the device name: "TestDevice":

AT+pINSDV=TestDevice:ADSerScaleType:-----:I-0:I-0

NOTE! The UC-321PL scale must be set to Weight A mode.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.4.2.1 A&D serial Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Device name> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.4.2.2 A&D serial Scale <deviceinfo> field

Parameter	**Value**
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with the device name: "TestDevice" and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum measurement limit in kg.
- GeneralScriptParameter2: Maximum measurement limit in kg.
- GeneralScriptParameter3: Target weight in kg.
- GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=TestDevice:ADSerScaleType:QT1,2---QT7–50–100–75-15

13.4.2.3 A&D serial Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970 (RTX337x time as the device has no internal clock)
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	-1 (not supported)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.4.3 Data to server

The <Monitor> part of the XML delivery format for the A&D Serial Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: “Ws1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

A&D Serial Scale:	
Configured to kg *<xml xmlns="RTX-H#1-Client">* *<Gateway>* *<GatewayId>00025B00A5ED</GatewayId>* *<Time>1159864615</Time>* *<MessageNumber>23</MessageNumber>* ***<DvType>ADSerScaleType</DvType>*** *<GwInfo>……..</GwInfo>* *<GwStatus>0</GwStatus>* *<GwChksum>……..</GwChksum>* *</Gateway>* *<Monitor>* ***<DeviceId>ADSER:23</DeviceId>*** *<Type>NORMAL</Type>* *<UtcTime>1156812754</UtcTime>* ***<Value>*** ***<SubDev>Ws1</SubDev>*** ***<SubValue>+084.00 kg</SubValue>*** ***</Value>*** *<DvChksum>……..</DvChksum>* *</Monitor>* *</xml>*	*Configured to lbs* *<xml xmlns="RTX-H#1-Client">* *<Gateway>* *<GatewayId>00025B00A5ED</GatewayId>* *<Time>1159864615</Time>* *<MessageNumber>23</MessageNumber>* ***<DvType>ADSerScaleType</DvType>*** *<GwInfo>……..</GwInfo>* *<GwStatus>0</GwStatus>* *<GwChksum>……..</GwChksum>* *</Gateway>* *<Monitor>* ***<DeviceId>ADSER:23</DeviceId>*** *<Type>NORMAL</Type>* *<UtcTime>1156812754</UtcTime>* ***<Value>*** ***<SubDev>Ws1</SubDev>*** ***<SubValue>+0132.0 lb</SubValue>*** ***</Value>*** *<DvChksum>……..</DvChksum>* *</Monitor>* *</xml>*

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>………</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>ADSER:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.5 THT Bluetooth weight scale

This section describes how to attach a THT Bluetooth weight scale to the RTX337x.

13.5.1 Compatible THT scales

- THT Bluetooth scale SC-1 and SC-2

13.5.2 Installing the THT Bluetooth weight scale

To install a THT weight scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The weight scale is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>*.***

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>.

<devicetype> for the THT Bluetooth weight scale is: **THTBTWSType**.

Below is an example on how to insert a THT weight scale with device name: "TestDevice" and the following Bluetooth address: 0007808125EA. Installation using the Bluetooth address prevents unauthorized connections to take place. The Bluetooth address is placed on the backside of the weight scale.

AT+pINSDV=TestDevice:THTBTWSType:0007808125EA-0289

Note: *This device cannot be inserted as a generic BT device because it is operating as a Bluetooth slave device so it has to be polled by the RTX337x and this requires the Bluetooth address to be known.*

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the folowing.
The JavaScripts will be activated with a set of parameters as described in the following.

13.5.2.1 THT Bluetooth Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.5.2.2 THT Bluetooth Scale <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"0289"
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with device name: "TestDevice" and the following Bluetooth address: 0007808125EA, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are not used:

AT+pINSDV=TestDevice:THTBTWSType:0007808125EA-0289-QT1,2---QT7

13.5.2.3 THT Bluetooth Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	-1 (not supported)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.5.3 Pairing with THT Bluetooth weight scale

After installing the weight scale on the RTX337x make a measurement on the weight scale. Paring should be made automatically.

13.5.4 Data to server

The <Monitor> part of the XML delivery format for the THT Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: “Ws1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

THT Bluetooth Scale:	
Configured to kg	*Configured to lbs*
<xml xmlns="RTX-H#1-Client">	*<xml xmlns="RTX-H#1-Client">*
<Gateway>	*<Gateway>*
<GatewayId>00025B00A5ED</GatewayId>	*<GatewayId>00025B00A5ED</GatewayId>*
<Time>1159864615</Time>	*<Time>1159864615</Time>*
<MessageNumber>23</MessageNumber>	*<MessageNumber>23</MessageNumber>*
<DvType>THTBTWSType</DvType>	***<DvType>THTBTWSType</DvType>***
<GwInfo>……..</GwInfo>	*<GwInfo>……..</GwInfo>*
<GwStatus>0</GwStatus>	*<GwStatus>0</GwStatus>*
<GwChksum>……..</GwChksum>	*<GwChksum>……..</GwChksum>*
</Gateway>	*</Gateway>*
<Monitor>	*<Monitor>*
<DeviceId>0007808125EA:23</DeviceId>	***<DeviceId>0007808125EA:23</DeviceId>***
<Type>NORMAL</Type>	*<Type>NORMAL</Type>*
<UtcTime>1156812754</UtcTime>	*<UtcTime>1156812754</UtcTime>*
<Value>	***<Value>***
<SubDev>Ws1</SubDev>	***<SubDev>Ws1</SubDev>***
<SubValue>+0084.0 kg</SubValue>	***<SubValue>+0132.0 lb</SubValue>***
</Value>	***</Value>***
<DvChksum>……..</DvChksum>	*<DvChksum>……..</DvChksum>*
</Monitor>	*</Monitor>*
</xml>	*</xml>*

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
    <Gateway>
        <GatewayId>00043EC204D2</GatewayId>
        <Time>………</Time>
        <MessageNumber>85</MessageNumber>
        <DvType>Gateway</DvType>
        <GwInfo>……….</GwInfo>
        <GwStatus>….</GwStatus>
        <GwChksum>……….</GwChksum>
    </Gateway>
    <Monitor>
        <DeviceId>0007808125EA:23</DeviceId>
        <Type>USERRESPONSE</Type>
        <Value>Text added by script</Value>
        <DvChksum>……..</DvChksum>
    </Monitor>
</xml>
```

13.5.4.1 Data port switch on scale

The THT scale has a “Data port” switch for settings of the output from the scale. Choose between ‘Kg’ and ‘lb’.

13.6 THT serial weight scale

This section describes how to attach a THT serial weight scale to the RTX337x.

13.6.1 Compatible THT scales

- SC-1 serial Telehealth Scale

13.6.2 Installing the THT serial weight scale

To install a THT weight scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The weight scale is installed using the **<devicetype>.**

The command for installing a serial THT weight scale is:
AT+pINSDV=<devicename>:<devicetype>:

<devicetype> for the THT Serial weight scale is: **THTSerScaleType**.

Below is an example on how to insert an A&D serial weight scale with the device name: "TestDevice":
AT+pINSDV=TestDevice:THTSerScaleType:-----:I-0:I-0

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.6.2.1 THT SC-1 Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Device name> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.6.2.2 THT SC-1 Scale <deviceinfo> field

Parameter	**Value**
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with the device name: "TestDevice" and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum measurement limit in kg.
- GeneralScriptParameter2: Maximum measurement limit in kg.
- GeneralScriptParameter3: Target weight in kg.
- GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=TestDevice:THTSerScaleType:QT1,2---QT7–50–100–75-15

13.6.2.3 THT SC-1 Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970 (RTX337x time as the device has no internal clock)
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	-1 (not supported)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.6.3 Data to server

The <Monitor> part of the XML delivery format for the THT SC-1 Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: “Ws1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

THT Serial Scale:	
Configured to kg *<xml xmlns="RTX-H#1-Client">* *<Gateway>* *<GatewayId>00025B00A5ED</GatewayId>* *<Time>1159864615</Time>* *<MessageNumber>23</MessageNumber>* ***<DvType>THTSerScaleType</DvType>*** *<GwInfo>……..</GwInfo>* *<GwStatus>0</GwStatus>* *<GwChksum>……..</GwChksum>* *</Gateway>* *<Monitor>* ***<DeviceId>THTScale:23</DeviceId>*** *<Type>NORMAL</Type>* *<UtcTime>1156812754</UtcTime>* ***<Value>*** ***<SubDev>Ws1</SubDev>*** ***<SubValue>+084.00 kg</SubValue>*** ***</Value>*** *<DvChksum>……..</DvChksum>* *</Monitor>* *</xml>*	*Configured to lbs* *<xml xmlns="RTX-H#1-Client">* *<Gateway>* *<GatewayId>00025B00A5ED</GatewayId>* *<Time>1159864615</Time>* *<MessageNumber>23</MessageNumber>* ***<DvType>THTSerScaleType</DvType>*** *<GwInfo>……..</GwInfo>* *<GwStatus>0</GwStatus>* *<GwChksum>……..</GwChksum>* *</Gateway>* *<Monitor>* ***<DeviceId>THTScale:23</DeviceId>*** *<Type>NORMAL</Type>* *<UtcTime>1156812754</UtcTime>* ***<Value>*** ***<SubDev>Ws1</SubDev>*** ***<SubValue>+0132.0 lb</SubValue>*** ***</Value>*** *<DvChksum>……..</DvChksum>* *</Monitor>* *</xml>*

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
    <Gateway>
        <GatewayId>00025B00A5ED</GatewayId>
        <Time>………</Time>
        <MessageNumber>85</MessageNumber>
        <DvType>Gateway</DvType>
        <GwInfo>……….</GwInfo>
        <GwStatus>….</GwStatus>
        <GwChksum>……….</GwChksum>
    </Gateway>
    <Monitor>
        <DeviceId>THTScale:23</DeviceId>
        <Type>USERRESPONSE</Type>
        <Value>Text added by script</Value>
        <DvChksum>……..</DvChksum>
    </Monitor>
</xml>
```

13.7 A&D Bluetooth Blood Pressure monitor

This section describes how to attach an A&D Bluetooth Blood Pressure monitor to the RTX337x.

13.7.1 Compatible A&D Blood Pressure monitors

- A&D Bluetooth Blood Pressure monitor UA-767 plus BT

13.7.2 Installing the A&D blood pressure monitor

To install a blood pressure monitor a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The blood pressure monitor is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>.**

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>.

<devicetype> for the A&D blood pressure monitor is: **ADBTBPType**.

Using Bluetooth address
Below is an example on how to insert an A&D Blood Pressure monitor with device name: “TestDevice” and the following Bluetooth address: 00043EC274D4. Installation using the Bluetooth address prevents unauthorized connections to take place. The Bluetooth address is placed on the backside of the blood pressure monitor.

AT+pINSDV=TestDevice:ADBTBPType:00043EC274D4-39121440

Generic installation
The BP monitor can also be inserted as generic, which opens the RTX337x for pairing with all A&D Bluetooth monitors, no matter which Bluetooth addresses they have. The device is only open for pairing with all monitors until the first successful connection. After this is completed the RTX337x will only respond to this monitor and is no longer considered generic.

AT+pINSDV=TestDevice:ADBTBPType:FFFFFFFFFFFF-39121440

NOTE: To prevent conflicts it is only possible to install one type of Bluetooth product as generic.

NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.7.2.1 A&D Blood Pressure Monitor Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.7.2.2 A&D Blood Pressure Monitor <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"39121440"
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the BPM with device name: "TestDevice" and the following Bluetooth address: 00043EC2044B (the Bluetooth address is placed on the back of the BPM).
And assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum systolic blood pressure.
- GeneralScriptParameter2: Maximum diastolic blood pressure.
- GeneralScriptParameter3: Minimum pulse.
- GeneralScriptParameter4: Maximum pulse.

AT+pINSDV=TestDevice:ADBTBPType:00043EC2044B-39121440-QT1,2---QT7-140-90-45-70

13.7.2.3 A&D Blood Pressure Monitor script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	SYS	Systolic value
1	DIA	Diastolic value
2	PUL	Pulse rate per minute
3	MAP	Mean Arterial Pressure
4	Measurement timestamp	Sec. since 1-Jan-1970 (If time is set to a date before production date this value will be -1)
5	BatteryStatus	0-255 (Conversion formula is: VBatt = value * 0.03V + 2.3V and the battery should be replaced when VBatt < 4.6V)
6	SaveStatus	1=success, 0=No save (buffer full).
7	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
8-14	:	
15	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.7.3 Pairing with A&D Blood Pressure monitor

To pair the Blood Pressure monitor with the RTX337x (which is prepared as described in section 13.7.2) some actions must be performed.

- Place the Blood Pressure monitor near the RTX337x (RTX337x must be powered).
- Make a measurement to initiate the data transmission.
- If the Blood Pressure monitor has already been paired to a RTX337x it will try to reconnect to that one.
- If the Blood Pressure monitor does not find the RTX337x it was already paired with, it will stop searching for 1 minute and then try again.
- If it then finds the RTX337x that it was already paired with, it delievers data.
- If it does not find the RTX337x that it was already paired with, it starts an inquiry to search for another device to pair with (any compatible Bluetooth device).
- If the Blood Pressure monitor finds a new compatible Bluetooth device, it pairs and delivers data.
- If the Blood Pressure monitor does not find a compatible Bluetooth device to pair with it stops searching and a new measurement must me made to restart the procedure.

Step by step installation

Step	Description	Tools
[1]	Configure the RTX337x. Check that the Blood Pressure monitor is inserted (AT+pINSDV?).	RTX337x + cable or web interface
[2]	Pair the Blood Pressure monitor to the RTX337x.	Blood pressure monitor and RTX337x
[3]	When a Blood Pressure monitor device is 'paired' and a measurement is made, the RTX337x tries to connect to the server and transmit data. If transmission fails, the data will be delivered next time the RTX337x connects to the server.	Blood pressure monitor and RTX337x

13.7.4 Data to server

The <Monitor> part of the XML delivery format for the A&D Bluetooth BMP includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>0

The format used for <Value> is: "Bpm1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

A&D Blood Pressure monitor:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>11</MessageNumber>
   <DvType>ADBTBPType</DvType>
   <GwInfo>…….. </GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>00043EC205BA:11</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus>100</BatteryStatus>
   <Value>
        <SubDev>Bpm1</SubDev>
        <SubValue>Sys:120 mmHg Dia:80  mmHg
           Pulse:70 1/min MAP:92 mmHg</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>………</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>00043EC205BA:11</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.8 IEM Bluetooth Blood Pressure monitor

This section describes how to attach an IEM Bluetooth Blood Pressure monitor to the RTX337x.

13.8.1 Compatible IEM Blood Pressure monitors

- IEM Bluetooth Blood Pressure monitor (Stabil-O-Graph)

13.8.2 Installing the IEM Blood Pressure monitor

To install a blood pressure monitor a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The blood pressure monitor is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>*.***

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>.

<devicetype> for the IEM blood pressure monitor is: **IEMBTBPType**.

Using Bluetooth address
Below is an example on how to insert an IEM blood pressure monitor with device name: "TestDevice" and the following Bluetooth address: 0006788556CF. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:IEMBTBPType:0006788556CF-6624
(The Bluetooth address is not labeled on the IEM Blood pressure monitor and if not known it may be replaced with FFFFFFFFFFFF for generic insertion.)

Generic installation
The BP monitor can also be inserted as generic, which opens the RTX337x for pairing with all IEM Bluetooth monitors, no matter which Bluetooth addresses they have. The device is only open for pairing with all monitors until the first successful connection. After this is completed the RTX337x will only respond to this monitor and is no longer considered generic.

AT+pINSDV=TestDevice:IEMBTBPType:FFFFFFFFFFFF-6624

NOTE: To prevent conflicts it is only possible to install one type of Bluetooth product as generic.

NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.8.2.1 IEM Blood Pressure Monitor Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.8.2.2 IEM Blood Pressure Monitor <deviceinfo> field

Parameter	Value
Bluetooth address	12 character hex (Capital letters only)
Pin code	"6624"
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the blood pressure monitor with device name: "TestDevice" and the following Bluetooth address: 0006788556CF and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum systolic blood pressure.
- GeneralScriptParameter2: Maximum diastolic blood pressure.
- GeneralScriptParameter3: Minimum pulse.
- GeneralScriptParameter4: Maximum pulse.

AT+pINSDV=TestDevice:IEMBTBPType:0006788556CF-6624-QT1,2---QT7-140-90-45-70

13.8.2.3 IEM Blood Pressure Monitor script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	SYS	Systolic value
1	DIA	Diastolic value
2	PUL	Pulse rate per minute
3	MAP	-1
4	Measurement timestamp	Sec. since 1-Jan-1970
5	TimeDifference	(IEM BPM time) – (RTX337x time) in seconds.
6	SaveStatus	1=success, 0=No save (buffer full).

7	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
8-12	:	
15	GeneralScriptParameter10	

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.8.3 Pairing with the IEM Blood Pressure monitor

To pair the blood pressure monitor with the RTX337x (which is prepared as described in section 13.7.2) some actions must be performed. Place the Blood pressure monitor near the RTX337x (RTX337x must be powered). Press the Menu button (first time long) until "PO2" flashes. Press Start/Stop button and "PO2" stops flashing. After a few seconds a short beep indicates a successful pairing, otherwise "Cod 01" is displayed and the pairing procedure must be repeated.

Step by step installation

Step	Description	Tools
[1]	Configure the RTX337x. Check that the blood pressure monitor is inserted (AT+pINSDV?).	RTX337x + cable or web interface
[2]	Pair the blood pressure monitor to the RTX337x.	Blood pressure monitor and RTX337x
[3]	When a blood pressure monitor device is 'paired' and a measurement is made, the RTX337x tries to connect to the server and transmit data. If transmission fails, the data will be delivered next time the RTX337x connects to the server.	Blood pressure monitor and RTX337x

13.8.4 Data to server

The <Monitor> part of the XML delivery format for the IEM Bluetooth BPM includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <TimeDiff>
- <Value>
- <DvChksum>

The format used for <Value> is: “Bpm1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

IEM Blood pressure monitor:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>12</MessageNumber>
   <DvType>IEMBTBPType</DvType>
   <GwInfo>…….. </GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>30100VXW:12</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
    <SubValue>Sys:134 mmHg Dia:081 mmHg
    Pulse:071 1/min MAP:000 mmHg</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>………</Time>
      <MessageNumber>15</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>30100VXW:12</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.9 A&D serial Blood Pressure monitor

This section describes how to attach an A&D serial Blood Pressure monitor to the RTX337x.

13.9.1 Compatible A&D Blood Pressure monitors

- A&D serial Blood Pressure monitor UA-767PC RS-232C

13.9.2 Installing the A&D blood pressure monitor

To install a blood pressure monitor a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The blood pressure monitor is installed using the **<devicetype>** and the ***<A&DbpmID>***.

The command for installing an A&D serial Blood Pressure monitor is:
AT+pINSDV=<devicename>:<devicetype>:*<A&DbpmID>:<Poll interval>*

<devicetype> for the A&D serial blood pressure monitor is: **ADSerBPMType**.

Using ***<A&DbpmID>***
If the ID in the Blood Pressure monitor on the first connection after the insertion does not match the ***<A&DbpmID>*** from the INSDV command, the ID in the specific Blood Pressure monitor is set to the INSDV command ***<A&DbpmID>*** and measurements (if any) in the Blood Pressure monitor are cleared. The ***<A&DbpmID>*** must be a 10 character string.

Poll interval
The communication is initiated by the RTX337x and is usually set to I-00.00.10.
The cable must be disconnected when no communication is needed. After a successful record transfer, a new measurement will not be transferred until the BPM has been disconnected for more than a poll period and then reconnected.

Below is an example on how to insert the A&D serial BPM with device name: "TestDevice", the following A&DbpmID: 0123456789 and Poll interval: "I-00.00.10".

AT+pINSDV=TestDevice:ADSerBPMType:0123456789:I-00.00.10

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.9.2.1 A&D serial Blood Pressure Monitor Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<A&DbpmID> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.

SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.9.2.2 A&D serial Blood Pressure Monitor <deviceinfo> field

Parameter	**Value**
A&DbpmID	ID. 10 ascii characters
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the A&D serial BPM with device name: "TestDevice" and the following A&DbpmID: 0123456789 and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned. In this example the General Parameters are not used.

AT+pINSDV=TestDevice:ADSerBPMType:0123456789-QT1,2---QT7:I-00.00.10

13.9.2.3 A&D serial Blood Pressure Monitor script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	SYS	Systolic value
1	DIA	Diastolic value
2	PUL	Pulse rate per minute
3	MAP	-1 (not used)
4	Measurement timestamp	Sec. since 1-Jan-1970
5	BatteryStatus	-1 (not used)
6	SaveStatus	1=success, 0=No save (buffer full).
7	Measurement index	Measurement index number in received list (starting with 1).
8	Total measurements	Total number of measurements in received list.
9	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
10-17	:	
18	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	

9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Total number of measurements in set	Integer
1	Measurements stored for upload	Integer
2	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
3-10	:	
11	GeneralScriptParameter10	

13.9.3 Data transfer with the A&D serial Blood Pressure monitor

The cable from the A&D serial Blood Pressure monitor must be disconnected when no communication is needed. After a successful record transfer, a new measurement will not be transferred until the BPM has been disconnected for more than a poll period and then reconnected.

13.9.4 Data to server

The <Monitor> part of the XML delivery format for the A&D Bluetooth BMP includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>0

The format used for <Value> is: "Bpm1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

A&D Blood Pressure monitor:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>11</MessageNumber>
   <DvType>ADSerBPMType</DvType>
   <GwInfo>........ </GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>........</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>07915:11</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>Bpm1</SubDev>
        <SubValue>Sys:120 mmHg Dia:080  mmHg
           Pulse:070 1/min MAP:103
           mmHg</SubValue>
   </Value>
   <DvChksum>........</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>.........</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>..........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>07915:11</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.10 Roche Blood Glucose meters and Coagulation monitor

This section describes how to attach a Roche Accu-Check Blood Glucose (BG) meters and CoaguChek coagulation monitor to the RTX337x*.*

13.10.1 Compatible Roche BG meters and Coagulation monitors

- Roche Accu-Chek Compact
  - name = **GCP**
- Roche Accu-Chek Compact Plus (Blue)
  - name = **GCP2**
- Roche Accu-Chek Compact Plus (Black)
  - name = **GCP2LCM**
- Roche Accu-Chek Aviva
  - name = **Aviva**
- Roche Accu-Chek Aviva (Serial number starting with 53)
  - name = **Aviva ASIC**
- Roche Accu-Chek Active
  - name = **ACCUCHEK Active**
- Roche Accu-Chek Performa
  - name = **Performa**
- Roche Accu-Chek Performa (Serial number starting with 55)
  - name = **Performa ASIC**
- Roche Accu-Chek Aviva Nano
  - name = **Mini 9**
- Roche CoaguChek XS
  - name = **CoaguChek Basic**

13.10.2 Installing the Roche Accu-Chek meters and CoaguChek

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation.
The meter is installed using the **<devicetype>** and the Accu-Chek or CoaguChek **name** and by option the last 8 digits of the **serial number*.*** The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing a Roche Accu-Chek meter or CoaguChek system is:
AT+pINSDV=<devicename>:<devicetype>:name,serial number;name,serial number:I-hh.mm.ss.

<devicetype> for Roche Accu-Chek meters and CoaguChek system is: **GenIR1Type** (No clock synchronisation) or **GenIR1TSType** (Clock synchronisation). Both Device types support all devices, however the time synchronisation is not supported for Accu-Chek Compact (GCP).

Note: *Time synchronisation will set the clock at the end of each communication session, this includes the situation where there are more measurements in the meter than the RTX337x can store.*

Using Serial number
Below is an example on how to insert a Roche Accu-Chek Compact Blood Glucose meter with device name: "TestDevice" and the following serial number:**07113409**. Installatior using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:GenIR1Type:GCP,07113409:I-00.00.10

Inserting Roche Accu-Chek Compact Plus and Aviva
AT+pINSDV=TestDevice:GenIR1Type:GCP2;01623069;Aviva,13920434:I-00.00.10

The Roche Accu-Chek meters and CoaguChek system can also be inserted generically, which opens the RTX337x for all compatible Roche Accu-Chek meters and CoaguChek systems no matter which serial number it has. To insert the Accu-Chek meter and CoaguChek system as generic the serial number is omitted.

Generic installation
AT+pINSDV=TestDevice:GenIR1Type:GCP:I-00.00.10

To insert Compact, Compact Plus, Aviva, Aviva ASIC and CoaguChek as generic:
AT+pINSDV=TestDevice:GenIR1Type:GCP;GCP2;Aviva;Aviva ASIC;CoaguChek Basic:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier below.
The JavaScripts will be activated with a set of parameters as described in the following.

13.10.2.1 Roche Accu-Chek and CoaguChek Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device type>	For each IR connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.10.2.2 Roche Accu-Chek and CoaguChek <deviceinfo> field

Parameter	Value
"ModelName,SerialNumber" pairs separated with ";"	• ModelName ○ "GCP" (Compact) ○ "GCP2" (Compact Plus - blue) ○ "GCP2LCM" (Compact Plus - black) ○ "Aviva" ○ "Aviva ASIC" (Serial number starting with 53) ○ "ACCUCHEK Active" ○ "Performa" ○ "Performa ASIC" (Serial number starting with 55) ○ "Mini 9" (Aviva Nano) ○ "CoaguChek Basic" (CoaguChek XS) • SerialNumber ○ The last 8 digits of the serial number on the back of the device. If less than 8 digits are present add left justified 0's.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Accu-Chek Compact with device name: "TestDevice" and the following serial number: 05629814 (the serial number is placed on the back of the device, note only the last 8 digits.) and an Aviva without any serial number, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompleted Event no JavaScript will be assigned. A poll interval of 10 seconds is also used.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Timeout limit of measurement.

AT+pINSDV=TestDevice:GenIR1Type:GCP,05629814;Aviva-QT1,2--QT7--5:I-00.00.10

13.10.2.3 Roche Accu-Chek script parameter map

MeasurementEvent(Accu-Chek):

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value.
1	Measurement (mmol/L)	The measurement value.
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit (for Aviva this is the value in the display measurements results are transferred as mg/dl)	0 = mg/dl, 1=mmol/L

4	Device type	-1 = Unknown, 0 = Compact, 1 = Compact Plus, 2 = Aviva, 3 = New Compact Plus, 4 = Performa, 5 = Active and 6 = Aviva Nano
5	Device Serial number	Integer (note, as a result of the conversion to double the serial number might be lacking left justified 0's)
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	Time difference	The difference in current time calculated as: DeviceTime – RTX337xtime
11	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
11-18	:	
20	GeneralScriptParameter10	

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Temperature out of range at measurement time.
Bit 1	Measurement marked as control level 1.
Bit 2	LO. Measurement below low setting on monitor.
Bit 3	HI. Measurement above high setting on monitor.
Bit 4	HYPO. Measurement below hypo setting on monitor. (Compact Plus, Aviva and Performa)
Bit 5	Data/time not set
Bit 6	Measurement marked as control level 2 (Aviva/Performa)
Bit 7	Used drum/barcode (Compact, Compact Plus and Active)
Bit 8	Expired drum/strip (for Roche Aviva this indicates strips expire in 30 days).
Bit 9	General Flag, Asterisk (Compact Plus & Aviva/Performa)
Bit 10	Result over personal target or control result above controls range (Aviva/Performa)
Bit 11	Result below personal target or control result below controls range (Aviva/Performa)
Bit 12	Control not identified (Aviva/Performa);
Bit 13-15	Unused

MeasurementEvent(CoaguChek):

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement (INR)	The measurement value.
1	Measurement (%Quick)	The measurement value.
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Display Units	1 = INR, 2=Quick, 3= sec (unit shown in the display of the CoaguChek)
4	Device type	9 = CoaguChek XS
5	Device Serial number	Integer (note, as a result of the conversion to double the serial number might be lacking left justified 0's)
6	Total number of measurements in set	Integer
7	Measurement number	Integer

8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	Time difference	The difference in current time calculated as: DeviceTime – RTX337xtime
11	Measurement (sec.)	The measurement value
12	INR high target	Current target value for INR in meter
13	INR low target	Current target value for INR in meter
14	Expiry date for test strip	Date in the format yymmdd (note, as a result of the conversion to double the serial number might be lacking left justified 0's)
15	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
15-23	:	
24	GeneralScriptParameter10	

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Date and time set by user.
Bit 1	Control measurement.
Bit 2	Range active (INR high/low check active)
Bit 3	Result of sec value is lower than measurement range.
Bit 4	Result of sec value is higher than measurement range
Bit 5	Result of INR value is lower than measurement range.
Bit 6	Result if INR value is higher than measurement range.
Bit 7	Result of %Quick value is lower than measurement range.
Bit 8	Result of %Quick value is higher than measurement range.
Bit 9-15	Unused

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Device type	-1= Unknown, 0 = Compact, 1 = Compact Plus and 2 = Aviva, 3 = New Compact Plus, 4 = Performa, 5 = Active, 6 = Aviva Nano, 9 = CoaguChek Basic
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	Time difference	The difference in current time calculated as: DeviceTime – RTX337xtime
4	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
5-12	:	

13.10.2.4 How to use Roche Accu-Check meters and CoaguChek monitor with the RTX337x

When the Accu-Chek meters and CoaguChek system are installed in the RTX337x the system is ready for use. The initiation of communication is dependent on which device is used.

Accu-Chek Compact: Place the Blood Glucose meter in front of the infrared window, open the lid and press the two small buttons located above the display ('MEM' and 'SET') simultaneously.

Accu-Chek Compact Plus: Place the Blood Glucose meter in front of the infrared window, press the 'M' and 'S' buttons simultaneously.

Accu-Chek Active: Place the Blood Glucose meter in front of the infrared window, press the 'M' button until "PC" flashes on the display.

Accu-Chek Aviva: Place the Blood Glucose meter in front of the infrared window, press the '◄' button and '►' button simultaneously until an arrow symbol appears.

*Accu-*Chek *Performa:* Place the Blood Glucose meter in front of the infrared window, press the '◄' button and '►' button simultaneously until an arrow symbol appears.

Accu-Chek Aviva Nano: Place the Blood Glucose meter in front of the infrared window, press the '◄' button and '►' button (on the top of the device) simultaneously until blinking arrow symbols appear in the display.

CoaguChek Basic (XS): Place the CoaguChek monitor in front of the infrared window (make sure the infrared windows are directly opposite each other). Turn on the CoaguChek and wait for the IR symbol to appear in the display of the CoaguChek.

13.10.3 Data to server

The <Monitor> part of the XML delivery format for the Roche Accu-Chek and CoaguChek devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <TimeDiff>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm1" for Accu-Chek meters and "Coagu1" for CoaguChek monitors.

Device type delivered to the Server is expanded with actual device type like this:
"GenIR1Type , Aviva"
The device types are:

Roche Accu-Chek Compact = GCP
Roche Accu-Chek Compact Plus (Blue) = GCP2
Roche Accu-Chek Compact Plus (Black) = GCP2LCM
Roche Accu-Chek Aviva = Aviva
Roche Accu-Chek Aviva (Serial number starting with 53) = Aviva ASIC

Roche Accu-Chek Active = ACCUCHEK Active
Roche Accu-Chek Performa = Performa
Roche Accu-Chek Performa (Serial number starting with 55) = Performa ASIC
Roche Accu-Chek Aviva Nano = Mini 9
Roche CoaguChek = CoaguChek Basic

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Roche Accu-Chek meter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>GenIR1Type</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>01623111:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <TimeDiff>0</TimeDiff>
   <Value>
        <SubDev>Bgm1</SubDev>
        <SubValue>00107 mg/dL  0000</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

Roche CoaguChek meter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>GenIR1Type</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>01623111:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <TimeDiff>0</TimeDiff>
<Value>
       <SubDev>Coagu1</SubDev>
       <SubValue>INR:000.9, %Quick:0111,
Sec:010.7, Unit:1, INR_L:001.5, INR_H:002.5,
MeasNr:00001, Lot:00005, Strip:15, Exp:051200,
Flag:0005, DateFmt:1, TimeFmt:1, Beeper:1,
LimitsOn:1 SWVer:03.02, HWVer:00000000000000,
Model:UP, MaxTrans:12, MaxMeas:100, Protocol:1.1,
MaxBytes:180, BootL:00.58, IRKey:IRKey, Bcycle:330,
PCBID:4104103001979P</SubValue>
   </Value>
 <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
     <GatewayId>00043EC204D2</GatewayId>
       <Time>………</Time>
       <MessageNumber>85</MessageNumber>
       <DvType>Gateway</DvType>
       <GwInfo>……….</GwInfo>
       <GwStatus>….</GwStatus>
       <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
       <DeviceId>01623111:23</DeviceId>
       <Type>USERRESPONSE</Type>
       <Value>Text added by script</Value>
       <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.11 LifeScan OneTouch Ultra Blood Glucose meter

This section describes how to attach a Lifescan OneTouch Ultra to the RTX337x.

13.11.1 Compatible LifeScan OneTouch Ultra meters

- LifeScan OneTouch Ultra (RS232)

13.11.2 Installing the Lifescan OneTouch Ultra meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The meter is installed using the **<devicetype>**, the blood glucose meters **serial number** and setting for display of units **x** (0=mg/dL, 1=mmol/L(default)) and setting of display of time format **y** (0=AM/PM, 1=24-hour). The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing a Lifescan OneTouch Ultra Blood Glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:serial number-x-y:I-hh.mm.ss

<devicetype> for the OneTouch Ultra blood glucose meter is: **LsBgmType**.

Using serial number
Below is an example on how to insert a LifeScan OneTouch Ultra Blood Glucose meter with device name: "TestDevice" and the following serial number: RST89A6QT.
Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:LsBgmType:RST89A6QT-0-1:I-00.00.10

Generic installation
The LifeScan OneTouch Ultra Blood Glucose meter can also be inserted as generic, which opens the RTX337x for all compatible LifeScan OneTouch Ultra Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 9*F.

AT+pINSDV=TestDevice:LsBgmType:FFFFFFFFF-0-1:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.11.2.1 Lifescan OneTouch Ultra Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.11.2.2 Lifescan OneTouch Ultra <deviceinfo> field

Parameter	Value
SerialNumber	SerialNumber ○ The 9 char serial number is placed on the back of the device.
Set display units	Glucose units setting on the meter display 0 = mg/dL (default) 1 = mmol/L
Set display time format	Time format on the meter display 0 = AM/PM format 1 = 24-hour format
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Lifescan OneTouch Ultra with device name: "TestDevice" and the following serial number: TQFC8DBGT (The serial number is placed on the back of the device), and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose value.
- GeneralScriptParameter2: Maximum blood glucose value.

AT+pINSDV=TestDevice:LsBgmType:TQFC8DBGT-0-0-QT1,2--QT7--80-120:I-00.00.10

13.11.2.3 Lifescan OneTouch Ultra script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value.
1	Measurement (mmol/L)	The measurement value.
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	0 = mg/dl, 1=mmol/L
4	Device type	10 = Lifescan OneTouch Ultra
5	Device Serial number	The Lifescan Onetouch Ultra always sends 0 as serial number. The device serial number is part of the activatorID, and could be read from JavaScripts using GetActivatorId().
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
11-18	:	
19	GeneralScriptParameter10	

Description of Status flag values.
All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Temperature out of range at measurement time.	(Not used)
Bit 1	Measurement marked as control level 1.	
Bit 2	LO. Measurement below low setting on monitor.	(Not used)
Bit 3	HI. Measurement above high setting on monitor.	
Bit 4	HYPO. Measurement below hypo setting on monitor.	(Not used)
Bit 5	Data/time not set	(Not used)
Bit 6	Measurement marked as control level 2	(Not used)
Bit 7	Used drum/barcode	(Not used)
Bit 8	Expired drum/strip	(Not used)
Bit 9	General Flag, Asterisk	(Not used)
Bit 10	Result over personal target	(Not used)
Bit 11	Result below personal target	(Not used)
Bit 12	Control not identified	(Not used)
Bit 13-15	Unused	(Not used)

Lifescan OneTouch Ultra only support bit 1 and bit 3.

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Device type	10 = Lifescan Onetouch Ultra
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.11.2.4 How to use LifeScan OneTouch Ultra meters with the RTX337x

When the Blood Glucose meter is installed in the RTX337x the system is ready for use. The standard LifeScan cable is plugged into the RTX337x.
The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the LifeScan Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in scripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.11.3 Data to server

The <Monitor> part of the XML delivery format for the Lifescan "OneTouch" devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

LifeScan OneTouch Ultra Blood glucose meter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>LsBgmType</DvType>
   <GwInfo>........</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>.........</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>RST89A6QT:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus></BatteryStatus>
   <Value>
        <SubDev>Bgm1</SubDev>
        <SubValue>00107 mg/dL  0000</SubValue>
   </Value>
   <DvChksum>.........</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>.........</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>..........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>RST89A6QT:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.12 LifeScan OneTouch Ultra 2 Blood Glucose meter

This section describes how to attach a Lifescan OneTouch Ultra 2 Blood Glucose meter to the RTX337x.

13.12.1 Compatible LifeScan OneTouch Ultra 2 meters

- LifeScan OneTouch Ultra 2 (RS232)

13.12.2 Installing the Lifescan OneTouch Ultra 2 meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The meter is installed using the **<devicetype>** and the blood glucose meters **serial number**. The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing a Lifescan OneTouch Ultra 2 Blood Glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:serial number:I-hh.mm.ss

<devicetype> for the OneTouch Ultra 2 blood glucose meter is: **LsBgm2Type**.

Using serial number
Below is an example on how to insert a LifeScan OneTouch Ultra 2 Blood Glucose meter with device name: "TestDevice" and the following serial number: WBX0E99BY.
Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:LsBgm2Type:WBX0E99BY:I-00.00.10

Generic installation
The LifeScan OneTouch Ultra 2 Blood Glucose meter can also be inserted as generic, which opens the RTX337x for all compatible LifeScan OneTouch Ultra 2 Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 9*F.

AT+pINSDV=TestDevice:LsBgm2Type:FFFFFFFFF:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.12.2.1 Lifescan OneTouch Ultra 2 Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.12.2.2 Lifescan OneTouch Ultra 2 <deviceinfo> field

Parameter	**Value**
SerialNumber	SerialNumber ○ The 9 char serial number is placed on the back of the device.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Lifescan OneTouch Ultra 2 with device name: "TestDevice" and the following serial number: WBX0E99BY (The serial number is placed on the back of the device), and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose value.
- GeneralScriptParameter2: Maximum blood glucose value.

AT+pINSDV=TestDevice:LsBgm2Type:WBX0E99BY-QT1,2--QT7--80-120:I-00.00.10

13.12.2.3 Lifescan OneTouch Ultra 2 script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value. (Might be 0 at some flag values)
1	Measurement (mmol/L)	The measurement value. (Might be 0 at some flag values)
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	0 = mg/dl, 1=mmol/L
4	Device type	20 = Lifescan OneTouch Ultra 2
5	Device Serial number	The Lifescan Onetouch Ultra 2 always sends 0 as serial number. The device serial number is part of the activatorID, and could be read from JavaScripts using GetActivatorId().
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	Meal flag	0=None, 1=Before meal, 2=After meal
11	Meal comment	0=No comment 1=Not enough food 2=Too much food 3=Mild exercise 4=Hard exercise 5=Medication 6=Stress 7=Illness 8=Feel hypo 9=Menses 10=Vacation 11=Other
12	TimeDifference	(Ultra2 time) – (RTX337x time) in seconds.
13	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
14-21	:	
22	GeneralScriptParameter10	

Description of Status flag values.
All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Temperature out of range at measurement time.	(Not used)
Bit 1	Measurement marked as control level 1.	
Bit 2	LO. Measurement below low setting on monitor.	(Not used)
Bit 3	HI. Measurement above high setting on monitor.	(Not used)
Bit 4	HYPO. Measurement below hypo setting on monitor.	(Not used)
Bit 5	Data/time not set	(Not used)
Bit 6	Measurement marked as control level 2	(Not used)
Bit 7	Used drum/barcode	(Not used)
Bit 8	Expired drum/strip	(Not used)
Bit 9	General Flag, Asterisk	(Not used)
Bit 10	Result over personal target	(Not used)
Bit 11	Result below personal target	(Not used)

Bit 12	Control not identified	(Not used)
Bit 13	Parity error	
Bit 14-15	Unused	(Not used)

Lifescan OneTouch Ultra 2 only support bit 1 and bit 13.

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Device type	20 = Lifescan Onetouch Ultra 2
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.12.2.4 How to use LifeScan OneTouch Ultra 2 meters with the RTX337x

When the Blood Glucose meter is installed in the RTX337x the system is ready for use. The standard LifeScan cable is plugged into the RTX337x.
The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the LifeScan Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in scripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.12.3 Data to server

The <Monitor> part of the XML delivery format for the Lifescan "OneTouch Ultra 2" devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <TimeDiff>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm2".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

LifeScan OneTouch Ultra 2 Blood glucose meter:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>.... </GwInfo>
    <DvType>LsBgm2Type</DvType>
    <GwChksum>…..</GwChksum>
    <Time>1188489713</Time>
    <MessageNumber>189</MessageNumber>
  </Gateway>
  <Monitor>
    <Value>
      <SubDev>Bgm2</SubDev>
      <SubValue>00128 mg/dL  0000 A 06</SubValue>
    </Value>
    <UtcTime>1188305121</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>2162691347</DvChksum>
    <DeviceId>WBX0E99BY:189</DeviceId>
  </Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <Time>…..</Time>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>…..</GwInfo>
    <GwChksum>…..</GwChksum>
    <DvType>Gateway</DvType>
    <MessageNumber>197</MessageNumber>
  </Gateway>
  <Monitor>
    <Value> Text added by script</Value>
    <Type>USERRESPONSE</Type>
    <DvChksum>……</DvChksum>
    <DeviceId>WBX0E99BY:189</DeviceId>
  </Monitor>
</xml>
```

13.13 LifeScan “OneTouch UltraEasy” / “OneTouch UltraMini” Blood Glucose meter

In this section the name “OneTouch UltraEasy” is used as reference to the device but it also covers the “OneTouch UltraMini” device.

This section describes how to attach a Lifescan OneTouch UltraEasy Blood Glucose meter to the RTX337x.

13.13.1 Compatible LifeScan OneTouch UltraEasy meters

- LifeScan OneTouch UltraEasy (RS232)
- LifeScan OneTouch UltraMini (RS232)

13.13.2 Installing the Lifescan OneTouch UltraEasy meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The meter is installed using the **<devicetype>,** the blood glucose meters **serial number** and select the Clear device memory option **x** (0 = Do not clear device memory, 1 = Clear device memory if all data is successfully transferred to RTX337x). The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing a Lifescan OneTouch UltraEasy Blood Glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:serial number-x:I-hh.mm.ss

<devicetype> for the OneTouch UltraEasy blood glucose meter is: **LsEasyType**.

Using serial number
Below is an example on how to insert a LifeScan OneTouch UltraEasy Blood Glucose meter with device name: “TestDevice”, the following serial number: WBX0E99BY and clearing device memory after successful transmission. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:LsEasyType:WBX0E99BY-1:I-00.00.10

Generic installation
The LifeScan OneTouch Ultra 2 Blood Glucose meter can also be inserted as generic, which opens the RTX337x for all compatible LifeScan OneTouch UltraEasy Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 9*F.

AT+pINSDV=TestDevice:LsEasyType:FFFFFFFFF-1:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is

"JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.13.2.1 Lifescan OneTouch UltraEasy Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.13.2.2 Lifescan OneTouch UltraEasy <deviceinfo> field

Parameter	**Value**
SerialNumber	SerialNumber ○ The 9 char serial number is placed on the back of the device.
Clear device memory	0 = Do not clear device memory 1 = Clear device memory if all data is successfully transferred to RTX337x
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Lifescan OneTouch UltraEasy with device name: "TestDevice" and the following serial number: WBX0E99BY (The serial number is placed on the back of the device), clearing device memory after successful transmission, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose value.
- GeneralScriptParameter2: Maximum blood glucose value.

AT+pINSDV=TestDevice:LsEasyType:WBX0E99BY-1-QT1,2--QT7--80-120:I-00.00.10

13.13.2.3 Lifescan OneTouch UltraEasy script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value. (Might be 0 at some flag values)
1	Measurement (mmol/L)	The measurement value. (Might be 0 at some flag values)
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	0 = mg/dl, 1=mmol/L
4	Device type	30 = Lifescan OneTouch UltraEasy
5	Device Serial number	The Lifescan Onetouch UltraEasy always sends 0 as serial number. The device serial number is part of the activatorID, and could be read from JavaScripts using GetActivatorId().
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	Not used.
9	SaveStatus	1=success, 0=No save (buffer full).
10	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
11-18	:	
19	GeneralScriptParameter10	

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Device type	30 = Lifescan Onetouch UltraEasy
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.13.2.4 How to use LifeScan OneTouch UltraEasy meters with the RTX337x

When the Blood Glucose meter is installed in the RTX337x the system is ready for use. The OneTouch UltraEasy meter has to be turned off to establish a connection with the RTX337x. Then the standard LifeScan cable is plugged into the RTX337x.

The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the LifeScan Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in scripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.13.3 Data to server

The <Monitor> part of the XML delivery format for the Lifescan "OneTouch UltraEasy" devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm1".
No status flags are used – all set inactive, "HI" and "LO" will not be used for the measurement value.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

LifeScan OneTouch UltraEasy Blood glucose meter:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>.... </GwInfo>
    <DvType>LsEasyType</DvType>
    <GwChksum>.....</GwChksum>
    <Time>1188489713</Time>
    <MessageNumber>189</MessageNumber>
  </Gateway>
  <Monitor>
    <Value>
      <SubDev>Bgm1</SubDev>
      <SubValue>00128 mg/dL  0000</SubValue>
    </Value>
    <UtcTime>1188305121</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>2162691347</DvChksum>
    <DeviceId>WBX0E99BY:189</DeviceId>
  </Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <Time>…..</Time>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>…..</GwInfo>
    <GwChksum>…..</GwChksum>
    <DvType>Gateway</DvType>
    <MessageNumber>197</MessageNumber>
  </Gateway>
  <Monitor>
    <Value> Text added by script</Value>
    <Type>USERRESPONSE</Type>
    <DvChksum>……</DvChksum>
    <DeviceId>WBX0E99BY:189</DeviceId>
  </Monitor>
</xml>
```

13.14 LifeScan "OneTouch Vita" Blood Glucose meter

In this section the name "OneTouch Vita" is used as reference to the device.

This section describes how to attach a Lifescan OneTouch Vita Blood Glucose meter to the RTX337x.

13.14.1 Compatible LifeScan OneTouch Vita meters

- LifeScan OneTouch Vita (RS232)

13.14.2 Installing the Lifescan OneTouch Vita meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The meter is installed using the **<devicetype>,** the blood glucose meters **serial number** and select the Clear device memory option **x** (0 = Do not clear device memory, 1 = Clear device memory if all data is successfully transferred to RTX337x). The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing a Lifescan OneTouch Vita Blood Glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:serial number-x:I-hh.mm.ss

<devicetype> for the OneTouch Vita blood glucose meter is: **LsVitaType**.

Using serial number
Below is an example on how to insert a LifeScan OneTouch Vita Blood Glucose meter with device name: "TestDevice", the following serial number: WBX0E99B and clearing device memory after successful transmission. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:LsVitaType:WBX0E99B-1:I-00.00.10

Generic installation
The LifeScan OneTouch Vita Blood Glucose meter can also be inserted as generic, which opens the RTX337x for all compatible LifeScan OneTouch Vita Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 8*F.

AT+pINSDV=TestDevice:LsVitaType:FFFFFFFF-1:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.14.2.1 Lifescan OneTouch Vita Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.14.2.2 Lifescan OneTouch Vita <deviceinfo> field

Parameter	**Value**
SerialNumber	SerialNumber ○ The 8 char serial number is placed on the back of the device.
Clear device memory	0 = Do not clear device memory 1 = Clear device memory if all data is successfully transferred to RTX337x
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Lifescan OneTouch Vita with device name: "TestDevice" and the following serial number: WBX0E99B (The serial number is placed on the back of the device), clearing device memory after successful transmission, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose value.
- GeneralScriptParameter2: Maximum blood glucose value.

AT+pINSDV=TestDevice:LsVitaType:WBX0E99B-1-QT1,2--QT7--80-120:I-00.00.10

13.14.2.3 Lifescan OneTouch Vita script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value. (Might be 0 at some flag values)
1	Measurement (mmol/L)	The measurement value. (Might be 0 at some flag values)
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	0 = mg/dl, 1=mmol/L
4	Device type	50 = Lifescan OneTouch Vita
5	Device Serial number	The Lifescan Onetouch Vita always sends 0 as serial number. The device serial number is part of the activatorID, and could be read from JavaScripts using GetActivatorId().
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	Only bit 1 is used: Measurement marked as control.
9	SaveStatus	1=success, 0=No save (buffer full).
10	Meal flag	0=None, 1=Before meal, 2=After meal, 3=Fasting
11	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
12-19	:	
20	GeneralScriptParameter10	

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Device type	50 = Lifescan Onetouch Vita
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.14.2.4 How to use LifeScan OneTouch Vita meters with the RTX337x

When the Blood Glucose meter is installed in the RTX337x the system is ready for use. The OneTouch Vita meter has to be turned off to establish a connection with the RTX337x. Then the standard LifeScan cable is plugged into the RTX337x.
The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the LifeScan Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in scripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.14.3 Data to server

The <Monitor> part of the XML delivery format for the Lifescan "OneTouch Vita" devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm2".
Only one status flag is used: "Measurement marked as control", "HI" and "LO" will not be used for the measurement value.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

LifeScan OneTouch Vita Blood glucose meter:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>.... </GwInfo>
    <DvType>LsVitaType</DvType>
    <GwChksum>.....</GwChksum>
    <Time>1188489713</Time>
    <MessageNumber>189</MessageNumber>
  </Gateway>
  <Monitor>
    <Value>
      <SubDev>Bgm2</SubDev>
      <SubValue>00128 mg/dL  0000 A 00</SubValue>
    </Value>
    <UtcTime>1188305121</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>2162691347</DvChksum>
    <DeviceId>WBX0E99B:189</DeviceId>
  </Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <Time>…..</Time>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>…..</GwInfo>
    <GwChksum>…..</GwChksum>
    <DvType>Gateway</DvType>
    <MessageNumber>197</MessageNumber>
  </Gateway>
  <Monitor>
    <Value> Text added by script</Value>
    <Type>USERRESPONSE</Type>
    <DvChksum>……</DvChksum>
    <DeviceId>WBX0E99B:189</DeviceId>
  </Monitor>
</xml>
```

13.15 LifeScan OneTouch UltraSmart Blood Glucose Meter

This section describes how to attach the Avita devices to the RTX337x.

13.15.1 Compatible LifeScan devices

- OneTouch UltraSmart

13.15.2 Installing the LifeScan OneTouch UltraSmart

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The meter is installed using the **<devicetype>** and the blood glucose meters **<serial number>**. The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing the device is:
AT+pINSDV=<devicename>:<devicetype>:<serial number>:I-hh.mm.ss

<devicetype> for the UltraSmart devices is: **LsSmartType**.

NOTE: Time synchronization will set the clock at each communication session, this includes the situation where there are more measurements in the meter than the RTX337x can store.

Using serial number
Below is an example on how to insert a LifeScan OneTouch UltraSmart Blood Glucose meter with device name: "TestDevice", the following serial number: ZCZ03CBNV and clearing device memory after successful transmission. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:LsSmartType:ZCZ03CBNV-1:I-00.00.10

Generic installation
The LifeScan OneTouch UltraSmart Blood Glucose meter can also be inserted as generic, which opens the RTX337x for all compatible LifeScan OneTouch UltraSmart Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 9*F.

AT+pINSDV=TestDevice:LsSmartType:FFFFFFFFF-1:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

NOTE: The LifeScan OneTouch UltraSmart measurement Limit for First connection refers to number of days back relative to first connection instead of numbers of measurements.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV –

Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.15.2.1 LifeScan OneTouch UltraSmart Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.15.2.2 LifeScan OneTouch UltraSmart <deviceinfo> field

Parameter	**Value**
SerialNumber	SerialNumber The 9 char serial number is placed on the back of the device.
Clear device memory	0 = Do not clear device memory 1 = Clear device memory if all data is successfully transferred to RTX337x
Set Beep on/off	0 = Beep off 1 = Beep on
Set Date format	0 = Date format: M.D.Y. 1 = Date format: D.M.Y.
Set Average time	0 = 7 day average 1 = 14 day average 2 = 30 day average 3 = 60 day average 4 = 90 day average
Set Start of week	0 = Week start: Monday 1 = Week start: Sunday
Set Language	0 = English 1 = Spanish 2 = French 3 = Italian 4 = Dutch 5 = Portuguese 7 = German 9 = Danish 10 = Norwegian 11 = Finnish
Set display time format	0 = AM/PM format 1 = 24-hour format
Set Cholesterol Unit	0 = mg/dL 1 = mmol/L

Set Insulin pump	0 = pump not used 1 = pump used
Set Meal schedule	0 = pre-set schedule 1 = personal schedule
Set Meal segments	The 7 segments with default times Before breakfast 6:00 – 9:00 After breakfast 9:00 – 11:00 Before Lunch 11:00 – 14:00 After Lunch 14:00 – 17:00 Before Dinner 17:00 – 20:00 After Dinner 20:00 – 23:00 Night 23:00 – 6:00 The values to be set are the time limit between the segments. The first value is the start-time of the before breakfast segment. All 7 values must contain valid times in 15 minute intervals since midnight (00 .. 95) otherwise the whole set is ignored. Use ‘,’ as separator. This is the string for the default values: 24,36,44,56,68,80,92
Assign insulin names	Up to 11, 8 character names for insulin types can be set. Use ‘,’ as separator.
Select insulin names	Up to 3 different insulin types can be selected. 01 Rapid 02 Regular 03 Lente 04 NPH 05 Ultralen 06 Premixed 07 Other 08 InsulinA 09 InsulinB 10 InsulinC 11 User defined 1 12 User defined 2 : 20 User defined 10 21 User defined 11 Use ‘,’ as separator.
Assign oral medication (pills) names	Up to 10, 8 character names for oral medication can be set. Use ‘,’ as separator.

Select oral medication (pills)	Up to 5 different pills can be selected. 01 Pill A 02 Pill B 03 Pill C 04 Pill D 05 Pill E 06 User defined 1 07 User defined 2 : 14 User defined 9 15 User defined 10 Use ‘,’ as separator.
Glucose limits	4 limit values between 060 and 200 mg/dL to be set: • Lower limit of glucose level before-meal • Upper limit of glucose level before-meal • Lower limit of glucose level after-meal • Upper limit of glucose level after-meal 1 limit value between 050 and 090 mg/dL to be set: • Hypo level Use ‘,’ as separator. Example: 065,180,070,170,060
MeasurementEvent	QuestionTreeIdentifier or nothing
SupervisionEvent	QuestionTreeIdentifier or nothing
ConnectionEvent	QuestionTreeIdentifier or nothing
ConnectionCompletedEvent	QuestionTreeIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of ‘.’ as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: “-” (0x2D)

NOTE: Insulin and oral medication names can only be changed if the name number is empty or the existing name is not represented in the UltraSmart log.

Below is an example on how to insert an Lifescan OneTouch UltraSmart with device name: “TestDevice” and the following configuration:

Serial number:	ZCZ03CBNV	
Clearing device memory:	on	
Beep:	on	
Date format:	M.D.Y.	
Average time:	30 days average	
Start of week:	Monday	
Language:	English	
Display time format:	24-hour	
Cholesterol Unit:	mg/dL	
Insulin pump:	pump not used	
Meal schedule:	personal schedule	
Meal segments:	Before breakfast	6:00 – 9:00
	After breakfast	9:00 – 11:00

	Before Lunch	11:00 – 14:00
	After Lunch	14:00 – 17:00
	Before Dinner	17:00 – 20:00
	After Dinner	20:00 – 23:00
	Night	23:00 – 6:00
Assign insulin names:	Lantus and Humalog	
Select insulin names:	NPH, Lantus and Humalog	
Assign oral medication names:	Actos	
Select oral medication:	Actos	
Glukose limits:	Lower limit of glucose level before-meal 65 mg/dL	
	Upper limit of glucose level before-meal 180 mg/dL	
	Lower limit of glucose level after-meal 70 mg/dL	
	Upper limit of glucose level after-meal 170 mg/dL	
	Hypo level 60 mg/dL	

And assigning a JavaScript named “QT1” for the Measurement Event at priority 2 and another JavaScript named “QT7” for the Connection Event at the default priority. For the EventRecordEvent, SupervisionEvent and ConnectionCompletedEvent no JavaScript will be assigned.

The poll interval is set to 10 seconds.

In this example the General Parameters are used like this: GeneralScriptParameter1: Minimum blood glucose value. GeneralScriptParameter2: Maximum blood glucose value.

```
AT+pINSDV=TestDevice:LsSmartType:ZCZ03CBNV-1-1-0-2-0-0-1-0-0-1-
24,36,44,56,68,80,92-Lantus,Humalog-04,11,12-Actos-06-
065,180,070,170,060-QT1,2---QT7-80-120:I-00.00.10
```

13.15.2.3 LifeScan OneTouch UltraSmart script parameter map

MeasurementEvent:

Index	Parameter	Format (all parameters are "doubles")
0	Device type	40 = Lifescan OneTouch UltraSmart
1	Record type	Glucose: 00 Glucose + comments Food: 20 Meal 1 21 Meal 2 Health: 30 Ketones 31 HbA1C 32 Micro albumin 33 Cholesterol LDL,HDL 34 Cholesterol Total, TG 35 Blood pressure 36 Weight, Height 37 Health notes 38 Last visit Medication: 10 Oral medication 11 Insulin injection 12 Setting Bolus 13 Setting Pump Daily Total Exercise: 45 Exercise
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Total number of measurements in set	Integer
4	Measurement number	Integer
5	Record Content	24 bits. See description below
6	SaveStatus	1=success, 0=No save (buffer full).
7	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
8-15	:	
16	GeneralScriptParameter10	

Description of Record Content.

Glucose Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
0	Glucose	Value: 0-1023 Commenting (see list of comments)	1	10 14	Meter always stores mg/dl.

Description of Comments.

Bit	Type		Value
Bit 10 (LSB)	Control solution	Blood (normal or alternate site),	0
		Control	1
Bit 11	Alternate site	Normal test (blood taken from finger tips)	0
		Alternate site test	1
Bit 12-14	Meal	No food commenting	0
		Before breakfast	1
		after breakfast	2
		Before lunch	3
		after lunch	4
		Before dinner	5
		after dinner	6
		Night	7
Bit 15	Reserved		-
Bit 16-17	Exercise	No exercise comment	0
		Before exercise	1
		During exercise	2
		After exercise	3
Bit 18	Health	Stress	0/1
Bit 19		Feel hypo	0/1
Bit 20		Illness	0/1
Bit 21		Menses	0/1
Bit 22		Vacation	0/1
Bit 23		Other	0/1

Food Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
20	Meal 1	Meal segment: brkf 0 lunch 1 dinner 2 snack 3 alc 4 Carbs: 0 . 250: 0 - 250 "---": 255 Fats: 0 . 250: 0 - 250 "---": 255	1 1 1	3 8 8 5 left	Carbs and fats are invalid if meal segment alcohol is set.

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
21	Meal 2 Protein	Meal segment: brkf 0 lunch 1 dinner 2 snack 3 alc 4 Cal: 0 - 2500: 0 - 500 "---": 511 Prot: 0 - 250: 0 - 250 "---": 255	1 5 1	3 9 8 4 left	Calories and proteins are invalid if meal segment alc is set.

Health Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
30	Ketones	neg. 0 trace 1 small 2 moderate 3 large 4	1	3 21 left	
31	HbA1C	4 - 15 % 40 -150	0.1 %	8 16 left	
32	Micro albumin	normal 0 positive 1 1 2 to 1000 1001 >1000 1002	1	10 14 left	
33	Cholesterol LDL, HDL	LDL: 0 - 500 0 - 500 >500 501 "---" 511 HDL: 0 - 500 0 - 500 >500 501 "---" 511	1 mg/dl 1 mg/dl	9 9 6 left	

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
34	Cholesterol Total, TG (Triglyceride)	Total: 0 - 1000 0-1000 >1000 1001 "---" 1023 TG: 0 - 3000 0-3000 >3000 3001 "---" 4095	1 mg/dl 1 mg/dl	10 12 2 left	Meter stores values in mg/dl but always displays in selected units Conversion between mg/dl and mmol/l for Total: Factor = 38,61 [6] Range 0 - >25.9 mmol/l 1001 mg/dl <> >25.9 mmol/l Conversion between mg/dl and mmol/l for TG: Factor = 88,5 [6] Range 0 – >33.9 mmol/l 3001 mg/dl <> >33.9 mmol/l
35	Blood pressure	80 - 200 80 - 200 40 - 150 40 - 150	1	9 9 6 left	
36	Weight height	Weight: 0 - 600 0 - 1200 "---" 2047 Height: 0 - 255 0 - 510 "---" 511	0.5 0.5	11 9 4 left	
37	Health notes	Stress Feel hypo Illness Menses Vacation Other	-	1 1 1 1 1 1 18 left	
38	Last visit	Doctor 0 Eye exam. 1 Foot exam. 2	1	2 22 left	

Medication Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
10	Oral medication pills intake	Pill type: 1 - 15 1 – 15 Number: 0 - 5 0 - 10	1 0.5	4 4 16 left	Type 0: None 5 names fixed (from language table), 10 names customizable via One Touch DMS.

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
11	Insulin injection	Insulin type: 1 - 21 1 – 21 Units: 0 - 250 0 - 2500	1 0.1	5 12 7 left	Type 0: None 10 names fixed (from language table), 11 names customizable via One Touch DMS.
12	Setting Bolus	Units: 0 - 25 0 -250	0.1	8 16 left	
13	Setting Pump Daily Total	Units: 0 - 100 0 -1000	0.1	10 14 left	

Exercise Records:

Number	Description	Range	Resolution	Used bits in Record data (LSB first)	Comment
45	Exercise	Level: mild 0 moderate 1 hard 2 Duration: 0 - 24h 0 - 288	1 5 min.	2 9 13 left	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Device type	40 = Lifescan Onetouch UltraSmart
1	Total number of measurements and event records in set	Integer
2	Measurements and event records stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.15.3 How to use LifeScan OneTouch UltraSmart with the RTX337x

When the Blood Glucose meter is installed in the RTX337x the system is ready for use. The OneTouch UltraSmart meter has to be turned on to establish a connection with the RTX337x. Then the standard LifeScan cable is plugged into the RTX337x.
The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the LifeScan Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in scripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.15.4 Data to server

The <Monitor> part of the XML delivery format for the LifeScan OneTouch UltraSmart device includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm3"

The <DeviceId> included in messages send from a JavaScript to the server and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number for each auto generated message to make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The measurement data that is delivered to the server is delivered in the format shown below *(Example)*.

LifeScan OneTouch UltraEasy Blood glucose meter:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B006398</GatewayId>
    <GwInfo>50100004, XX, </GwInfo>
    <DvType>LsSmartType</DvType>
    <GwChksum>2277313964</GwChksum>
    <Time>1228747276</Time>
    <MessageNumber>770</MessageNumber>
  </Gateway>
  <Monitor>
    <Value>
      <SubDev>Bgm3</SubDev>
      <SubValue>00 000000</SubValue>
    </Value>
    <UtcTime>1228747276</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>1916800685</DvChksum>
    <DeviceId>ZCZ03CBNV:770</DeviceId>
  </Monitor>
</xml>
```

The measurement data telegrams can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts, see description of ClearResp(), AddToResp() and SendRespToServer() in the Technical Reference Manual. The secondary data telegrams have the exact same identification (<DeviceId>) as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00087B006398</GatewayId>
      <Time>………</Time>
      <MessageNumber>773</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>ZCZ03CBNV:770</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.16 Abbot/TheraSense blood glucose meters

This section describes how to attach an Abbot/TheraSense Blood Glucose meter to the RTX337x.

13.16.1 Compatible Abbot/TheraSense Blood Glucose meters

- Abbot/TheraSense FreeStyle (RS232)
- Abbot/TheraSense FreeStyle Flash (RS232)
- Abbot/TheraSense FreeStyle Lite (RS232)
- Abbot/TheraSense FreeStyle Freedom Lite (RS232)

13.16.2 Installing the Abbot/TheraSense Blood Glucose meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The monitor is installed using the **<devicetype>** and the Blood Glucose meters ***<serial number>.*** The poll interval is set for each device type in the format**: I-hh.mm.ss**.

The command for installing an Abbot/TheraSense blood glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:<serial number>:I-hh.mm.ss

<devicetype> for the TheraSense blood glucose meter is: **FreeStyleType**.

Using Serial number
Below is an example on how to insert an Abbot/ThreaSense Blood Glucose meter with device name: "TestDevice" and the following serial number: T-F230-62265. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:FreeStyleType:T-F230-62265:I-00.00.10

Generic installation
The Abbot/TheraSense Blood glucose meters can also be inserted as generic, which opens the RTX337x for all Abbot/Therasense FreeStyle and FreeStyle Flash Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meter as generic the serial number is replaced with 12*F.

AT+pINSDV=TestDevice:FreeStyleType:FFFFFFFFFFFF:I-00.00.10

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier below.

The JavaScripts will be activated with a set of parameters as described in the following.

13.16.2.1 Abbot/Therasense Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.16.2.2 Abbot/Therasense <deviceinfo> field

Parameter	Value
SerialNumber	SerialNumber ○ The serial number is placed on the back of the device.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Abbot/Therasense Freestyle with device name: "TestDevice" and the following serial number: T-G328-61571 (the Serial number is placed on the back of the device) and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds, and in the last example, supervision timing is set to 24 hour.

In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose.
- GeneralScriptParameter2: Maximum blood glucose.

AT+pINSDV=TestDevice:FreeStyleType:T-G328-61571-QT1,2--QT7--80-120:I-00.00.10

13.16.2.3 Abbot/Therasense script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value.
1	Measurement (mmol/L)	The measurement value.
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	This value is always 0, because there is no possibility to read this information out of the device.
4	Device type	11 = Abbot/Therasense Freestyle/Freestyle Flash/Freestyle Lite/Freestyle Freedom Lite
5	Device Serial number	The Abbot/Therasense Freestyle devices always send 0 as serial number. The device serial number is part of the activatorID, and could be read from scripts using GetActivatorId().
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
11-18	:	
19	GeneralScriptParameter10	

Description of Status flag values.
All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Temperature out of range at measurement time.	
Bit 1	Measurement marked as control level 1.	
Bit 2	LO. Measurement below low setting on monitor.	
Bit 3	HI. Measurement above high setting on monitor.	
Bit 4	HYPO. Measurement below hypo setting on monitor.	(Not used)
Bit 5	Data/time not set	
Bit 6	Measurement marked as control level 2	(Not used)
Bit 7	Used drum/barcode	(Not used)
Bit 8	Expired drum/strip	(Not used)
Bit 9	General Flag, Asterisk	(Not used)
Bit 10	Result over personal target	(Not used)
Bit 11	Result below personal target	(Not used)
Bit 12	Control not identified	(Not used)
Bit 13-15	Unused	(Not used)

Freestyle types only support bit 0,1,2,3 and bit 5.

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")

0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Device type	11 = Abbot/Therasense "Freestyle"
1	Total number of measurements in set	Integer
2	Measurements stored for upload	Integer
3	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
4-11	:	
12	GeneralScriptParameter10	

13.16.2.4 How to use Abbot/TheraSense Blood Glucose meters with the RTX337x

When the Blood glucose meter is installed in the RTX337x the system is ready for use.
The standard Abbot/TheraSense FreeStyle cable is plugged into the RTX337x.
The RTX337x tries to communicate with the Blood Glucose meter with intervals as specified in the DV-Timing parameter. Communication is indicated by the display in the Abbot/TheraSense Blood Glucose meter. When data has been transmitted (can be indicated by the RTX337x if implemented in JavaScripts) the cable must be unplugged from the Blood Glucose meter to prevent the battery in the Blood Glucose meter to be drained.

13.16.3 Data to server

The <Monitor> part of the XML delivery format for the Abbot/Therasense "Freestyle" devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

TheraSense Blood Glucose meter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>FreeStyleType</DvType>
   <GwInfo>........</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>........</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>T-F230-62265:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus></BatteryStatus>
   <Value>
        <SubDev>Bgm1</SubDev>
        <SubValue>00107 mg/dL
0000</SubValue>
   </Value>
   <DvChksum>........</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00043EC204D2</GatewayId>
      <Time>.........</Time>
      <MessageNumber>27</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>..........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>T-F230-62265:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.17 Bayer Serial Blood Glucose Meters

This section describes how to attach a Bayer Serial Blood Glucose meter to the RTX337x.

13.17.1 Compatible Bayer Blood Glucose meters

- BREEZE® (RS232)
- BREEZE2® (RS232)
- CONTOUR® (15- and 5-seconds version) (RS232)

13.17.2 Installing the Bayer Serial Blood Glucose meter

To install an external device a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each Blood Glucose meter inserted a new device name must be selected.
The monitor is installed using the **<devicetype>**, **<SubDevicetype>** and the Blood Glucose meters **<serial number>.**

The command for installing an Bayer Serial blood glucose meter is:
AT+pINSDV=<devicename>:<devicetype>:<SubDevicetype>,<serial-number>

<devicetype> for the Bayer Serial blood glucose meter is: **BayerSerBgmType.**

<SubDevicetype> for the Bayer Serial blood glucose meter are: **BREEZE, BREEZE2, CONTOUR5** and **CONTOUR15.**

Using Serial number
Below is an example on how to insert an Bayer Serial Blood Glucose meter with device name: "TestDevice" and the following serial number: 4679775. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:BayerSerBgmType:BREEZE,4679775

It is possible to insert more specific types but only one of each type like this:

**AT+pINSDV=TestDevice:BayerSerBgmType:BREEZE,4679775;
BREEZE2,4612375;CONTOUR5,4587123;CONTOUR15,1237123**

Generic installation
The Bayer Serial Blood glucose meters can also be inserted as generic, which opens the RTX337x for all supported Bayer Serial Blood Glucose meters no matter which serial number they have. To insert the Blood Glucose meters as generic the serial number is omitted.

AT+pINSDV=TestDevice:BayerSerBgmType:BREEZE;BREEZE2

It is possible to limit the number of measurements received at first connection by setting the parameter with the AT+DVLIMITFIRST – Set or request Device measurement Limit for First connection command or adding it to the AT+pINSDV – Insert or List Devices command. See details in section 9: AT+p Command Set.

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these

AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier below.
The JavaScripts will be activated with a set of parameters as described in the following.

13.17.2.1 Bayer Serial Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Serial number> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each successful RS232 connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For each successful measurement download a ConnectionCompletedEvent is generated

13.17.2.2 Bayer Serial <deviceinfo> field

Parameter	**Value**
"ModelName,SerialNumber" pairs separated with ";"	• ModelName ○ "BREEZE" ○ "BREEZE2" ○ "CONTOUR5" ○ "CONTOUR15" • SerialNumber BREEZE® Meters ○ Only digits after the hyphen in the serial number on the back of the device. • SerialNumber CONTOUR™ Meters ○ All digits of the serial number on the back of the device. If the serial number is omitted all of the specified model can connect.
Buzzer Enable	0 = disabled, 1 = enabled (Not changed if empty)
Buzzer Level	BREEZE® meters only 0 = Low, 1 = High (Not changed if empty)
Time Format	0 = 12 hr, 1 = 24 hr (Not changed if empty)
Glucose Units	0 = mg/dL, 1 = mmol/L (Not changed if empty) No longer supported for CONTOUR™ Meters.
Date Format	0 = month-day, 1 = day-month (Not changed if empty)
Test Time Alarm and Result Markers	CONTOUR™ 5-second Meter only No longer supported.
Reference Method	CONTOUR™ Meters only No longer supported.
Calibration Curve	CONTOUR™ Meters only No longer supported.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator.

:	These parameters are for general use in the JavaScripts.
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert an Bayer serial Blood Glucose Meter with device name: "TestDevice" and the following serial number: 4679775 (the Serial number is placed on the back of the device) and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Event at the default priority. For the Supervision Event and ConnectionCompletedEvent no JavaScript will be assigned.
The poll interval is set to 10 seconds, and in the last example, supervision timing is set to 24 hour.

In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum blood glucose.
- GeneralScriptParameter2: Maximum blood glucose.

AT+pINSDV=TestDevice:BayerSerBgmType:CONTOUR5,4679775;BREEZE2,4679123---1-0-----QT1,2--QT7--80-120

13.17.2.3 Bayer Serial script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Measurement (mg/dL)	The measurement value.
1	Measurement (mmol/L)	The measurement value.
2	Measurement Timestamp	Sec. since 1-Jan-1970
3	Measurement Unit in display	0 = mg/dl, 1=mmol/L
4	Device type	1 = BREEZE® 2 = BREEZE®2 3 = CONTOUR™ (15 sec.) 4 = CONTOUR™ (5 sec.)
5	Device Serial number	The serial number
6	Total number of measurements in set	Integer
7	Measurement number	Integer
8	Status Flag	See description below
9	SaveStatus	1=success, 0=No save (buffer full).
10	Meal flag	0=None, 1=Before meal, 2=After meal 3=Logbook
11	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
12-19	:	
20	GeneralScriptParameter10	

Description of Status flag values.
All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Temperature out of range at measurement time.
Bit 1	Measurement marked as control level 1. (Not used)
Bit 2	LO. Measurement below low setting on monitor.
Bit 3	HI. Measurement above high setting on monitor.
Bit 4	Measurement marked as User-marked control
Bit 5	Measurement marked as User-deleted result
Bit 6	Measurement marked as Self detected control
Bit 7-12	Unused
Bit 13	Parity error (Not used)
Bit 14-15	Unused

Bayer serial Blood Glucose Meter do not support bit 1 and bit 7-15.

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	14 days average Measurement (mg/dL)	The measurement value. (0 indicates no average available)
1	14 days average Measurement (mmol/L)	The measurement value. (0 indicates no average available)
2	Measurement Unit in display	0 = mg/dl, 1=mmol/L
3	Device type	1 = BREEZE® 2 = BREEZE®2 3 = CONTOUR™ (15 sec.) 4 = CONTOUR™ (5 sec.)
4	Device Serial number	The serial number
5	Total number of measurements in set	Integer
6	Measurements stored for upload	Integer
7	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
8-15	:	
16	GeneralScriptParameter10	

13.17.2.4 How to use Bayer Serial Blood Glucose meters with the RTX337x

When the Blood glucose meter is installed in the RTX337x the system is ready for use. When the standard Bayer Serial Blood Glucose meter cable is plugged into the RTX337x and the Bayer Device. Then activate the Bayer Serial Blood Glucose meter and the RTX337x tries to communicate with the Blood Glucose meter. Communication is indicated by the display in the Bayer Serial Blood Glucose meter.

13.17.3 Data to server

The <Monitor> part of the XML delivery format for the Bayer Serial Blood Glucose meter devices includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bgm2".

Device type delivered to the Server is expanded with actual device type like this:
"BayerSerBgmType,CONTOUR5"
The device types are:

BREEZE = BREEZE®
BREEZE2 = BREEZE® 2
CONTOUR15 = CONTOUR™ (15 sec.)
CONTOUR5 = CONTOUR™ (5 sec.)

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Bayer serial Blood Glucose Meter:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>.... </GwInfo>
    <DvType>BayerSerBgmType,CONTOUR5</DvType>
    <GwChksum>…..</GwChksum>
    <Time>1188489713</Time>
    <MessageNumber>189</MessageNumber>
  </Gateway>
  <Monitor>
    <Value>
      <SubDev>Bgm2</SubDev>
      <SubValue>00128 mg/dL  0000 N 00</SubValue>
    </Value>
    <UtcTime>1188305121</UtcTime>
    <Type>NORMAL</Type>
    <DvChksum>2162691347</DvChksum>
    <DeviceId>4679775:189</DeviceId>
  </Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
  <Gateway>
    <GwStatus>0</GwStatus>
    <Time>…..</Time>
    <GatewayId>00087B00638B</GatewayId>
    <GwInfo>…..</GwInfo>
    <GwChksum>…..</GwChksum>
    <DvType>Gateway</DvType>
    <MessageNumber>197</MessageNumber>
  </Gateway>
  <Monitor>
    <Value> Text added by script</Value>
    <Type>USERRESPONSE</Type>
    <DvChksum>……</DvChksum>
    <DeviceId>4679775:189</DeviceId>
  </Monitor>
</xml>
```

13.18 ViaSys AM1 Bluetooth Asthma monitor

This section describes how to attach a ViaSys AM1 Bluetooth Asthma monitor to the RTX337x.

13.18.1 Compatible ViaSys AM1 Asthma monitors

- ViaSys AM1(+) Bluetooth Asthma monitor

13.18.2 Installing the ViaSys Bluetooth Asthma monitor

To install a ViaSys Asthma monitor a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The Asthma monitor is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>** and parameter settings.

The command for installing a Bluetooth device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the ViaSys Asthma monitor is: **ViasysAM1BTType**.

Using Bluetooth address
Below is an example on how to insert a ViaSys Asthma monitor with device name: “TestDevice” and the following Bluetooth address: 00A09616502. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:ViasysAM1BTType:00A096161502-1609----24.00;24.00;24.00;24.00;24.00----1

Note: *This device cannot be inserted as a generic BT device because it is operating as a Bluetooth slave device so it has to be polled by the RTX337x and this requires the Bluetooth address to be known.*

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.18.2.1 ViaSys Bluetooth Asthma monitor Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
KeyEvent	<Bluetooth address> :<MessageNumber>	For each received patient key a KeyEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For every successful measurement a ConnectionCompletedEvent is generated.

13.18.2.2 ViaSys Bluetooth Asthma monitor <deviceinfo> field

Parameter	Value
Bluetooth address	12 character hex (Capital letters only)
Pin code	“1609”
Bluetooth Friendly name	(currently not used)
Patient Identification	Identifier for patient maximum 20 characters.
Display parameter	Which parameter to display after a measurement session. 1 = PEF and 2 = FEV1 If empty the setting in the meter will not be changed.
Alarm hours	5 alarms (time of day) in 10 minute steps can be configured as “hour.minute” pairs separated with “;” unused times should be “24.00”. If empty the setting in the meter will not be changed.
Threshold Yellow – Red	Threshold value for the displayed parameter. For PEF this is a value between 101 and 999 liters/minute For FEV this is a value between 200 and 9940 ml If empty the setting in the meter will not be changed. This is only set if both Display and Green – Yellow is non empty.
Threshold Green – Yellow	Threshold value for the displayed parameter. For PEF this is a value between 101 and 999 liters/minute For FEV this is a value between 200 and 9940 ml If empty the setting in the meter will not be changed. This is only set if both Display and Yellow – Red is non empty.

Variability	Controls whether the daily variability is shown after a measurement: 0 = Variability deactivated. 1 = Variability activated. If empty the setting in the meter will not be changed.
Best Measurement	Determines whether the device selects the best measurement within 10 minutes for storage or stores all measurements. The parameter which is displayed is the one used in this selection. 0 = activated 1 = deactivated If empty the setting in the meter will not be changed.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptTreeIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
KeyEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Asthma monitor with device name: "TestDevice" and the following Bluetooth address: 00043EC204D2, no settings changed, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event, the Connection Event and the Key Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Upper PEF limit
- GeneralScriptParameter2: Lower PEF limit

AT+pINSDV=TestDevice:ViasysAM1BTType:00043EC204D2-1609---------QT1,2---QT7–-400–300

13.18.2.3 ViaSys Bluetooth Asthma monitor script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	PEF Measurement (litre/min)	The measureement value. (value might be -1 if that particular measurement value was unavailable)
1	FEV1 Measurement (litre)	The measurement value. (value might be -1 if that particular measurement value was unavailable)
2	FVC Measurement (litre)	The measurement value. (value might be -1 if that particular measurement value was unavailable)

3	FEF75 Measurement (litre/s)	The measurement value. (value might be -1 if that particular measurement value was unavailable)
4	FEF50 Measurement (litre/s)	The measurement value. (value might be -1i f that particular measurement value was unavailable)
5	FEF25 Measurement (litre/s)	The measurement value. (value might be -1 if that particular measurement value was unavailable)
6	FEF 75/25 (litre/s)	The measurement value. (value might be -1 if that particular measurement value was unavailable)
7	Measurement Timestamp	Sec. since 1-Jan-1970
8	Measurement mode	0 = Pre measurement 1 = Measurement medication 2 = Measurement events 3 = Measurement symptoms
9	Total number of measurements in set	Integer
10	Measurement number	Integer
11	SaveStatus	1=success, 0=No save (buffer full).
12	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
13-20	:	
21	GeneralScriptParameter10	

KeyEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Key value	The measurement value, the format is: abcd, where 'a' indicates type (1 = Medication, 2= Event, 3= Symptoms). For Medication and Event 'b' is the value 0-3 and c, d are always 0. For Symptoms 'b' = Couch value 0-3, 'c' Dyspnea value 0-3 and 'd' Sputum value 0-3.
1	Key Timestamp	Sec. Since 1-Jan-1970
2	Total number of keys in set	Integer
3	Key number	Integer
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	

9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Total number of measurements in set	Integer
1	Measurements stored for upload	Integer
2	Total number of Keys in set	Integer
3	Keys stored for upload	Integer
4	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
5-12	:	
13	GeneralScriptParameter10	

13.18.2.4 How to use ViaSys Bluetooth Asthma monitor with the RTX337x

When the Viasys Asthma monitor is installed in the RTX337x the system is ready for use. The RTX337x tries to communicate with the Viasys Asthma monitor with intervals. It is important that the Viasys Asthma monitor is not paired with the RTX337x or any other device manually; the RTX337x will look for and connect to the Viasys Asthma monitor with the specified Bluetooth address on its own.
When measurement upload should be performed enter the menu on the asthma monitor.

- Select the rightmost icon '•))|((•'.
- Press the ok button (green dot). This should make the icon change with a '√' in the one corner.
- Wait for the Viasys Asthma monitor to be found by the RTX337x.

If the Viasys Asthma monitor has been paired with another device this can be cleared by (with Viasys Asthma monitor off) simultaneously pressing and releasing right and left key, then select ESC twice (left key) then select the rightmost icon and press OK (should show an X as part of the icon at the end).

13.18.3 Data to server

The <Monitor> part of the XML delivery format for the Viasys AM1 Bluetooth includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The formats used for <Value> is: "Spiro-1" for the measurement message and "AM1key-1" for the key message.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

ViaSys AM1(+) Asthma monitor:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>ViasysAM1BTType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>08876290AB15:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>Spiro-1</SubDev>
        <SubValue>FVC: xxxx ml, PEF: xxx l/min,  FEV1:
xxxx ml/s, FEF75: xxxx ml/s, FEF50: xxxx ml/s,
FEF25: xxxx ml/s, FEF75_25: xxxx ml/s, Mode:
c</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

For key events the <SubDev> value field is "AM1Key-1" and the <SubValue> value field is "Key: abcd" where 'a' is the type (*medication, event* or *symptoms*). For *medication* and *event* 'b' is the value (0-3), 'c' and 'd' are empty. For *symptoms* 'b' is the cough value (0-3), 'c' is the dyspnea value (0-3) and 'd' is the Sputum value (0-3).

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>………</Time>
      <MessageNumber>25</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>08876290AB15:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.19 MIR Spirotel spirometer

This section describes how to attach a MIR spirotel spirometer to the RTX337x.

13.19.1 Compatible MIR spirometers

- MIR Spirotel spirometer (RS232 cable)

13.19.2 Installing the MIR spirotel spirometer

To install a MIR spirotel a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The MIR spirotel is installed using the **<devicetype>** and the **<Serial number>** and parameter settings.

The command for installing a MIR device is:
AT+pINSDV=<devicename>:<devicetype>:<Serial number>----------------

<devicetype> for the MIR spirotel is: **MirSpirotelType**.

Using serial number
Below is an example on how to insert a MIR Spirotel with device name: "TestDevice" and the following serial number: 123456. Installation using the serial number prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:MirSpirotelType:123456----------------

Generic installation
The MIR Spirotel spirometer can also be inserted as generic, which opens the RTX337x for all compatible MIR Spirotel spirometers no matter which serial number they have. To insert the Spirometer as generic the serial number is replaced with 6*F.

AT+pINSDV=TestDevice:MirSpirotelType:FFFFFF----------------

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.19.2.1 MIR spirotel Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
SpirometerMeasurementEvent	<Serial number> :<MessageNumber>	For each received Spirometer measurement a MeasurementEvent is generated.
OximeterMeasurementEvent	<Serial number> :<MessageNumber>	For each received Oximeter measurement an OximeterMeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Serial number>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Serial number>	For every successful measurement a ConnectionCompletedEvent is generated.

13.19.2.2 MIR Spirotel <deviceinfo> field

Parameter	**Value**
Serial number	6 ascii characters
Traffic light curve type	Decimal number: 0 = Off (No traffic light) 1 = FVC 2 = FEV1 3 = PEF 4 = FEF2575 Illegal or empty value is set to 0 (Default)
Traffic light reference value	Decimal number: FVC (0-700 cL) FEV1 (0-700 cL) PEF (0-1200 L/min) FEF2575 (0-1000 cL/s) Value > 80% of ref. Green light 50% < Value < 80% of ref. Yellow light Value < 50% of ref. Red light Illegal or empty value sets traffic light curve type to 0 (No Traffic light)

Symptoms and Questions	Decimal number: If bit n = 0 no question is asked on that issue. Bit 0=1: Breathing difficulty? Bit 1=1: Chest Tightness? Bit 2=1: Cough? Bit 3=1: Sputum production? Bit 4=1: Disturbed sleep? Bit 5=1: Wheezing? Bit 6=1: (Not used) Bit 7=1: (Not used) Bit 8=1: Is it a week day? Bit 9=1: Drug taken? (Not working properly for software version: 3.4 RC) Bit 10=1: (Not used) Bit 11=1: (Not used) Illegal or empty value sets traffic light curve type to 0 (No questions asked)
Age	Age in years Set to 50 if illegal or empty value
Height	Height in cm. Set to 180 if illegal or empty value
Weight	Weight in kg. Set to 80 if illegal or empty value
Gender	0 = Male (Default) 1 = Female
Values format	0 = Normal value 1 = Calculated value using Age, Height, Weight and Gender using Knudson equations (Age, Height, Weight and Gender must be set).
Show values in percent	0 = Do not use percentage 1 = Show percentage of the calculated value (Age, Height, Weight and Gender must be set).
Timestep	Time interval between one data saving and the next one. 4 = 4 seconds 8 = 8 seconds 12 = 12 seconds (Default) 20 = 20 seconds If empty or illegal value is specified, the default value is set.
BeepOnOff	Set warning beeper on or off. 0 = Off (Default) 1 = On. If empty or illegal value is specified, the default value is set.
SpO2Min	Minimum value for SpO2 warning. Range: 70 – 100. Illegal or empty value is set to 80 (Default)

SpO2Max	Maximum value for SpO2 warning. Range: 70 – 100. Illegal or empty value is set to 100 (Default)
HRMin	Minimum value for Heart rate warning. Range: 30 – 250. Illegal or empty value is set to 40 (Default)
HRMax	Maximum value for Heart rate warning. Range: 30 – 250. Illegal or empty value is set to 180 (Default)
Use Info button	Use Info button RTX337x to trigger measurement reading. 0 = No 1 = Yes
SpirometerMeasurementEvent	JavaScriptIdentifier or nothing
OximeterMeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptTreeIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
KeyEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the MIR Spirotel with the following settings:
device name: "TestDevice"
Serial number: 204711
FEV1 to be used for the traffic light: 1
Reference value for the traffic light: 500
Ask for Chest Tightness and if Drug is taken: 514
Do not use calculated values and percentage showing of values.
Enable the Info button on the RTX337x to be used to activate a measurement reading.

AT+pINSDV=TestDevice:MirSpirotelType:204711-1-500-514-------------1

13.19.2.3 MIR Spirotel script parameter map

SpirometerMeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	FVC	FVC value in cL.
1	FEV1	FEV1 value in cL.
2	FEV1%	(FEV1/FVC*100) in percent
3	PEF	PEF in L/min
4	FEF2575	FEF2575 in cL/s
5	FET	FET in 1/10s

6	SYMPTOM	Answers to the 8 symptom questions. One digit to each symptom.
		Symptom1=Breathing difficulty
		Symptom2=Chest Tightness
		Symptom3=Cough
		Symptom4=Sputum production
		Symptom5=Disturbed sleep
		Symptom6=Wheezing
		Symptom7=(Not used)
		Symptom8=(Not used)
		First digit to Symptom1, next to Symptom2 and so on.
		Format:
		0 = Not activated
		1 = No
		2 = Average
		
		Example: 3200000
		Chest Tightness=Maximum.
		Cough=Average.
		NOTE: Zeros before first non-zero digit is not shown.

7	QUESTION	Answers to the 4 questions. One digit to each question. Question1=Is it a week day Question2=Drug taken (Not working properly for software version: 3.4 RC) Question3=(Not used) Question4=(Not used) Format: 0 = Not activated 1 = Yes 2 = No Example: 2000 Is it a week day=No NOTE: Zeros before first non-zero digit is not shown.
8	QUALITY	The binary interpretation of the bits set in the value: Bit0=No problem Bit1=Repeat test and start faster Bit2=Repeat test without coughing Bit3=Breathe out for a longer time Bit4=Breathe out all air in the lungs Bit5=Warning! Values dropping Bit6=Warning! There's variability Bit7=The spirometry is reproducible
9	Measurement timestamp	Sec. since 1-Jan-1970
10	BatteryStatus	0=0% charge 1=25% charge 2=50% charge 3=75% charge 4=100% charge
11	SaveStatus	1=success, 0=No save (buffer full).

12	Measurement index	Measurement index number in received list (starting with 1).
13	Total measurements	Total number of measurements in received list.
14	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
15-22	:	
23	GeneralScriptParameter10	

OximeterMeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	HRMin	Minimum heart rate during measurement
1	HRMax	Maximum heart rate during measurement
2	HRAvg	Average heart rate during measurement
3	SpO2Min	Minimum SpO2 value during measurement
4	SpO2Max	Maximum SpO2 value during measurement
5	SpO2Avg	Average SpO2 value during measurement
6	SYMPTOM	Answers to the 8 symptom questions. One digit to each symptom. Symptom1=Breathing difficulty Symptom2=Chest Tightness Symptom3=Cough Symptom4=Sputum production Symptom5=Disturbed sleep Symptom6=Wheezing Symptom7=(Not used) Symptom8=(Not used) First digit to Symptom1, next to Symptom2 and so on. Format: 0 = Not activated 1 = No 2 = Average 3 = Maximum Example: 203000 Cough=Average. Disturbed sleep=Maximum. NOTE: Zeros before first non-zero digit is not shown.

7	QUESTION	Answers to the 4 questions. One digit to each question. Question1=Is it a week day Question2=Drug taken (Not working properly for software version: 3.4 RC) Question3=(Not used) Question4=(Not used) Format: 0 = Not activated 1 = Yes 2 = No Example: 2000 Is it a week day=No. NOTE: Zeros before first non-zero digit is not shown.
8	Measurement timestamp	Sec. since 1-Jan-1970
9	BatteryStatus	0=0% charge 1=25% charge 2=50% charge 3=75% charge 4=100% charge
10	SaveStatus	1=success, 0=No save (buffer full).
11	Measurement index	Measurement index number in received list (starting with 1).
12	Total measurements	Total number of measurements in received list.
13	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
14-21	:	
22	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Total number of Spirometer measurements in set	Integer
1	Spirometer measurements stored for upload	Integer
2	Total number of Oximeter measurements in set	Integer
3	Oximeter measurements stored for upload	Integer
4	GeneralScriptParameter1	These parameters are for general use in

5-12	:	the JavaScripts.
13	GeneralScriptParameter10	

13.19.2.4 How to use MIR Spirometer with the RTX337x

When the MIR spirometer is installed in the RTX337x and the cable is plugged into the RTX337x the system is ready for use. The MIR spirometer has to polled by the RTX337x in order to deliver data. This can be done via script or by the event of pressing the 'I' button. The latter requires that the 'use info button' parameter is set, see section 13.19.2.2.

13.19.3 Data to server

The <Monitor> part of the XML delivery format for the MIR oth includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The formats used for <Value> is: "Spiro-2" for the Spirometer measurement message and "SpO2-2" for the Oximeter measurement message.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

MIR Spirometer, Spirometer measurement:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>MirSpirotelType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>204711:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Batterystatus>3<BatteryStatus>
   <Value>
        <SubDev>Spiro-2</SubDev>
        <SubValue>FVC: xxxx cL, FEV1: xxxx cL,
FEV1%: xxx, PEF: xxxx L/min, FEF2575: xxxx cL/s,
FET: xxxx 1/10s, SYMPTOM: xxxxxxxx, QUESTION:
xxxx, QUALITY:  01, SW: xxxx</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

MIR Spirometer, Oximeter measurement:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>3</MessageNumber>
   <DvType>MirSpirotelType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>204711:3</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Batterystatus>3<BatteryStatus>
   <Value>
        <SubDev>SpO2-2</SubDev>
        <SubValue>HRMin: xxx bpm, HRMax: xxx bpm,
HRAvg: xxx bpm, SpO2Min: xxx %, SpO2Max: xxx %,
SpO2Avg: xxx %, SYMPTOM: xxxxxxxx, QUESTION:
xxxx, QUALITY:  01, SW: xxxx</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>………</Time>
      <MessageNumber>25</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>……….</GwInfo>
      <GwStatus>….</GwStatus>
      <GwChksum>……….</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>08876290AB15:23</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>……..</DvChksum>
   </Monitor>
</xml>
```

13.20 Nonin Ipod serial Oximeter

This section describes how to attach a Nonin Ipod serial oximeter to the RTX337x.

13.20.1 Compatible Nonin serial oximeters

- Nonin Ipod 3212 serial oximeter

13.20.2 Installing the Nonin Ipod serial oximeter

To install a Nonin Ipod serial oximeter a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected. The oximeter is installed using the **<devicetype>*.***

The command for installing the serial Nonin Ipod oximeter is:
AT+pINSDV=<devicename>:<devicetype>:-----

<devicetype> for the Nonin Ipod oximeter is: **NoninIpodType**

Below is an example on how to insert a Nonin Ipod serial oximeter with device name: "TestDevice":
AT+pINSDV=TestDevice:NoninIpodType:-----

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.20.2.1 Nonin Ipod Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Device name> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.20.2.2 Nonin Ipod <deviceinfo> field

Parameter	**Value**
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Nonin Ipod with device name: "TestDevice" and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Minimum Heart rate limit beats per minute.
- GeneralScriptParameter2: Maximum Heart rate limit beats per minute.
- GeneralScriptParameter3: Target SpO_2
- GeneralScriptParameter4: SpO_2 spread

AT+pINSDV=TestDevice:NoninIpodType:QT1,2---QT7–40.0–80.0-97-3

13.20.2.3 Nonin ipod script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Heart rate Measurement	The measurement value (4 beat average).
1	SpO2 Measurement	The measurement value (4 beat average).
2	Measurement Timestamp	Sec. Since 1-Jan-1970 (RTX337x time as the device has no internal clock)
3	Status Flag	See description below
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag never raise this flag.

Bit 0	Green perfusion
Bit 1	Yellow perfusion
Bit 2	Red perfusion
Bit 3-15	Unused

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.20.3 Data to server

The <Monitor> part of the XML delivery format for the Nonin Ipod includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "SpO2-1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Nonin Ipod Oximeter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>82</MessageNumber>
   <DvType>NoninIpodType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>devicename:82</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:072 bpm SpO2:097 %
0001</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
    <Gateway>
        <GatewayId>00025B00A5ED</GatewayId>
        <Time>.........</Time>
        <MessageNumber>85</MessageNumber>
        <DvType>Gateway</DvType>
        <GwInfo>.........</GwInfo>
        <GwStatus>....</GwStatus>
        <GwChksum>.........</GwChksum>
    </Gateway>
    <Monitor>
        <DeviceId>devicename:82</DeviceId>
        <Type>USERRESPONSE</Type>
        <Value>Text added by script</Value>
        <DvChksum>.......</DvChksum>
    </Monitor>
</xml>
```

13.21 Nonin 4100BT Oximeter

This section describes how to attach a Nonin 4100BT Oximeter to the RTX337x.

13.21.1 Compatible Nonin Oximeters

- Nonin 4100BT Oximeter with 8000AA series sensor

13.21.2 Installing the Nonin 4100BT Oximeter

To install a Nonin 4100BT oximeter a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The Nonin 4100BT is installed using the **<devicetype>**, the **<Bluetooth address>**, the Bluetooth **<Pin code>** and parameter settings.

The command for installing a Nonin 4100BT device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the Nonin 4100BT oximeter is: **Nonin4100BTType**.

Using Bluetooth address
Below is an example on how to insert a Nonin 4100BT with device name: "TestDevice" and the following Bluetooth address: 123456123456. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:Nonin4100BTType:123456123456-123456--------

Note: *This device cannot be inserted as a generic BT device because it is operating as a Bluetooth slave device so it has to be polled by the RTX337x and this requires the Bluetooth address to be known.*

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.21.2.1 Nonin 4100BT oximeter Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For every successful measurement a ConnectionCompletedEvent is generated.

13.21.2.2 Nonin 4100BT oximeter <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	Last 6 characters of the device serial number.
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptTreeIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Nonin 4100BT with device name: "TestDevice" and the following Bluetooth address: 123456123456, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum Heart rate limit beats per minute
- GeneralScriptParameter2: Minimum Heart rate limit beats per minute
- GeneralScriptParameter3: Target SpO2
- GeneralScriptParameter4: SpO2 spread

AT+pINSDV=TestDevice:Nonin4100BTType:123456123456-123456-QT1,2---QT7–80.0–40.0-97-3

13.21.2.3 Nonin 4100BT oximeter script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Heart rate Measurement	The measurement value (4 beat average).
1	SpO2 Measurement	The measurement value (4 beat average).
2	Measurement Timestamp	Sec. Since 1-Jan-1970 (RTX337x time as the device has no internal clock)
3	Status Flag	See description below
4	SaveStatus	1=success, 0=No save (buffer full).
5	BatteryStatus	0 = OK, 1 = Low (less than 30 minutes)
6	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
7-14	:	
15	GeneralScriptParameter10	

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag, never raise this flag.

Bit 0	Green perfusion (not used)
Bit 1	Yellow perfusion
Bit 2	Red perfusion
Bit 3	Artifact
Bit 4-15	Unused

SupervisionEvent:

Index	Parameter	Format (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	Format (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	Format (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.21.3 Data to server

The <Monitor> part of the XML delivery format for the Nonin 4100BT includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: "SpO2-1"

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Nonin 4100BT oximeter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>Nonin4100BTType</DvType>
   <GwInfo>........</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>........</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>123456123456:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus>0<BatteryStatus>
   <Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:072 bpm SpO2:097 %
0001</SubValue>
   </Value>
   <DvChksum>........</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
    <Gateway>
        <GatewayId>00025B00A5ED</GatewayId>
        <Time>.........</Time>
        <MessageNumber>25</MessageNumber>
        <DvType>Gateway</DvType>
        <GwInfo>..........</GwInfo>
        <GwStatus>....</GwStatus>
        <GwChksum>..........</GwChksum>
    </Gateway>
    <Monitor>
        <DeviceId>123456123456:23</DeviceId>
        <Type>USERRESPONSE</Type>
        <Value>Text added by script</Value>
        <DvChksum>........</DvChksum>
    </Monitor>
</xml>
```

13.22 Nonin 9560 OnyxII oximeter

This section describes how to attach a Nonin 9560 OnyxII Oximeter to the RTX337x.

13.22.1 Compatible Nonin Oximeters

- Nonin 9560 OnyxII Bluetooth oximeter

13.22.2 Installing the Nonin 9560 OnyxII BT Oximeter

To install a Nonin 9560 OnyxII BT oximeter a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The Nonin 9560 OnyxII is installed using the **<devicetype>**, the **<Bluetooth address>**, the Bluetooth **<Pin code>** and parameter settings.

The command for installing a Nonin 9560 OnyxII BT device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the Nonin 9560 OnyxII BT oximeter is: **NoninOnyx2BTType**.
<Pin code> for the Nonin 9560 OnyxII BT oximeter is the last 6 digits in the serial number.

Using Bluetooth address
Below is an example on how to insert a Nonin 9560 OnyxII BT with device name: "TestDevice" and the following Bluetooth address: 123456123456. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:NoninOnyx2BTType:123456123456-123456--------

Note: *This device cannot be inserted as a generic BT device because it is operating as a Bluetooth slave device so it has to be polled by the RTX337x and this requires the Bluetooth address to be known.*

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.22.2.1 Nonin 9560 OnyxII BT oximeter Events

Event	Activator ID / Server <DeviceId>	Comment
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For every successful measurement a ConnectionCompletedEvent is generated.

13.22.2.2 Nonin 9560 OnyxII BT oximeter <deviceinfo> field

Parameter	Value
Bluetooth address	12 character hex (Capital letters only)
Pin code	Six digit Serial number
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptTreeIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Nonin 9560 OnyxII with device name: "TestDevice" and the following Bluetooth address: 123456123456, and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum Heart rate limit beats per minute
- GeneralScriptParameter2: Minimum Heart rate limit beats per minute
- GeneralScriptParameter3: Target SpO2
- GeneralScriptParameter4: SpO2 spread

AT+pINSDV=TestDevice:NoninOnyx2BTType:123456123456-123456-QT1,2---QT7–80.0–40.0-97-3

13.22.2.3 Nonin 9560 OnyxII BT oximeter script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Heart rate Measurement	The measurement value (4 beat average).
1	SpO2 Measurement	The measurement value (4 beat average).
2	Measurement Timestamp	Sec. Since 1-Jan-1970 (RTX337x time as the device has no internal clock)
3	Status Flag	See description below
4	SaveStatus	1=success, 0=No save (buffer full).
5	BatteryStatus	0 = OK, 1 = Low (less than 30 minutes)

6	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
7-14	:	
15	GeneralScriptParameter10	

Description of Status flag values, names in parenthesis indicate that only the named devices support the specified flag. All flags are active 1 and devices that do not support a given flag, never raise this flag.

Bit 0	Green perfusion (not used)
Bit 1	Yellow perfusion
Bit 2	Red perfusion
Bit 3	Artifact
Bit 4-15	Unused

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.22.3 How to use Nonin 9560 Onyx II with the RTX337X

Once the specific Onyx II has been configured in the RTX337X put it on the finger (this turns the device on) and wait for the measurement to stabilize and be transmitted. If nothing happens please verify the Bluetooth address and PIN code are both correct. The connection may take a minute (more if there are a multiple Slave Bluetooth devices configured in the RTX337X).

Note: It is important to turn the 9560 Onyx II device on after changing the batteries (it is not required to take a measurements, just open the device so the display turns on then wait for it to turn off), as the device is otherwise left in a state where it listens to incoming Bluetooth connections, but never transmits data, this drains the battery and blocks other Bluetooth devices from connecting to the RTX337X

13.22.4 Data to server

The <Monitor> part of the XML delivery format for the Nonin 9560 OnyxII BT includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>

- <DvChksum>

The format used for <Value> is: "SpO2-1"

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Nonin 9560 OnyxII BT oximeter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>23</MessageNumber>
   <DvType>NoninOnyx2BTType</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>123456123456:23</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <BatteryStatus>0<BatteryStatus>
   <Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:071 bpm SpO2:098 %
0000</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:
<xml xmlns="RTX-H#1-Client"> *<Gateway>* *<GatewayId>00025B00A5ED</GatewayId>* *<Time>………</Time>* *<MessageNumber>25</MessageNumber>* ***<DvType>Gateway</DvType>*** *<GwInfo>……….</GwInfo>* *<GwStatus>….</GwStatus>* *<GwChksum>……….</GwChksum>* *</Gateway>* ***<Monitor>*** ***<DeviceId>123456123456:23</DeviceId>*** ***<Type>USERRESPONSE</Type>*** ***<Value>Text added by script</Value>*** *<DvChksum>……..</DvChksum>* ***</Monitor>*** *</xml>*

13.23 Mindray PM50 Oximeter

This section describes how to attach a Mindray PM50 oximeter to the RTX337x.

13.23.1 Compatible Mindray PM50 oximeters

- Mindray PM50 (Serial)

13.23.2 Installing the Mindray PM50 serial oximeter

To install a Mindray PM50 oximeter a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected. The oximeter is installed using the **<devicetype>*.***

The command for installing the serial Mindray PM50 oximeter is:
AT+pINSDV=<devicename>:<devicetype>:-----

<devicetype> for the Mindray PM50 oximeter is: **MindrayPM50Type**

Below is an example on how to insert a Mindray PM50 oximeter with device name: "TestDevice":
AT+pINSDV=TestDevice:MindrayPM50Type:-----

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.23.2.1 Mindray PM50 Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Device name> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.23.2.2 Mindray PM50 <deviceinfo> field

Parameter	Value
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' As separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Mindray PM50 with device name: "TestDevice" and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum Heart rate limit beats per minute.
- GeneralScriptParameter2: Minimum Heart rate limit beats per minute.
- GeneralScriptParameter3: Target SpO_2

AT+pINSDV=TestDevice:MindrayPM50Type:QT1,2---QT7–80.0–40.0-97

13.23.2.3 Mindray PM50 script parameter map

MeasurementEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	Heart rate Measurement	The measurement value (4 beat average).
1	SpO2 Measurement	The measurement value (4 beat average).
2	Measurement Timestamp	Sec. Since 1-Jan-1970 (RTX337x time as the device has no internal clock)
3	Status Flag	0 (not used)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	Parameter	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in

1-8	:	the JavaScripts.
9	GeneralScriptParameter10	

13.23.3 Data to server

The <Monitor> part of the XML delivery format for the Mindray PM50 includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: “SpO2-1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Mindray PM50 Oximeter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>82</MessageNumber>
   <DvType>MindrayPM50Type</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>devicename:82</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:072 bpm SpO2:097 %
0000</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>.........</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>.........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>devicename:82</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.24 Mindray PM60 Oximeter

This section describes how to attach a Mindray PM60 oximeter to the RTX337x.

13.24.1 Compatible Mindray PM60 oximeters

- Mindray PM60 (Infrared)

13.24.2 Installing the Nonin Ipod serial oximeter

To install a Mindray PM60 oximeter a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected. The oximeter is installed using the **<devicetype>*.***

The command for installing the serial Mindray PM60 oximeter is:
AT+pINSDV=<devicename>:<devicetype>:-----

<devicetype> for the Mindray PM60 oximeter is: **MindrayPM60Type**

Below is an example on how to insert a Mindray PM60 oximeter with device name: “TestDevice”:
AT+pINSDV=TestDevice:MindrayPM60Type:-----

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.24.2.1 Mindray PM60 Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Device name> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Device name>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Device name>	For each successful measurement download a ConnectionCompletedEvent is generated

13.24.2.2 Mindray PM60 <deviceinfo> field

Parameter	**Value**
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of ‘.’ As

:	separator.
GeneralScriptParameter10	These parameters are for general use in the JavaScripts.

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Mindray PM60 with device name: "TestDevice" and assigning a JavaScript named "QT1" for the Measurement Event at priority 2 and another JavaScript named "QT7" for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.

In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum Heart rate limit beats per minute.
- GeneralScriptParameter2: Minimum Heart rate limit beats per minute.
- GeneralScriptParameter3: Target SpO_2

AT+pINSDV=TestDevice:MindrayPM60Type:QT1,2---QT7–80.0–40.0-97

13.24.2.3 Mindray PM60 script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Heart rate Measurement	The measurement value (4 beat average).
1	SpO2 Measurement	The measurement value (4 beat average).
2	Measurement Timestamp	Sec. Since 1-Jan-1970 (RTX337x time as the device has no internal clock)
3	Status Flag	0 (not used)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.24.3 Data to server

The <Monitor> part of the XML delivery format for the Mindray PM60 includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <Value>
- <DvChksum>

The format used for <Value> is: "SpO2-1".

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Mindray PM60 Oximeter:

```
<xml xmlns="RTX-H#1-Client">
<Gateway>
   <GatewayId>00025B00A5ED</GatewayId>
   <Time>1159864615</Time>
   <MessageNumber>82</MessageNumber>
   <DvType>MindrayPM60Type</DvType>
   <GwInfo>……..</GwInfo>
   <GwStatus>0</GwStatus>
   <GwChksum>……..</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>devicename:82</DeviceId>
   <Type>NORMAL</Type>
   <UtcTime>1156812754</UtcTime>
   <Value>
        <SubDev>SpO2-1</SubDev>
        <SubValue>HR:072 bpm SpO2:097 %
0000</SubValue>
   </Value>
   <DvChksum>……..</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts (see 10.8.18, ClearResp(), 10.8.19, AddToResp() and 10.8.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00025B00A5ED</GatewayId>
      <Time>.........</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>.........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>devicename:82</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.25 Corscience BT 3/6 and BT 12 ECG recorders

This section describes how to attach a Corscience ECG recorder to the RTX337x.

13.25.1 Compatible ECG recorders

- Corscience BT 3/6 Bluetooth ECG Recorder
- Corscience BT 12 Bluetooth ECG Recorder

13.25.2 Installing the Corscience ECG Recorders

To install a Corscience ECG Bluetooth Recorder (BT 3/6 and BT 12) a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The ECG Recorder is installed using the **<devicetype>**, the **<Bluetooth address>** and the Bluetooth **<Pin code>*.***

The command for installing a Bluetooth ECG Recorder is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>.

<devicetype> for the Corscience Bluetooth ECG Recorder is: **CS3612ECGType**.

Using Bluetooth address
Below is an example on how to insert a Corscience ECG Recorder with device name: “TestDevice” and the following Bluetooth address: 00043EC204D4. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice:CS3612ECGType:00043EC204C2-1234

Note: *This device cannot be inserted as a generic BT device because it is operating as a Bluetooth slave device so it has to be polled by the RTX337x and this requires the Bluetooth address to be known*

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or list devices or AT+pUPDDV – Update device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is “JavaScriptName,priority”. If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the folowing.
The JavaScripts will be activated with a set of parameters as described in the following.

13.25.2.1 Corscience ECG Recorders Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.25.2.2 Corscience ECG Recorders <deviceinfo> field

Corscience ECG Bluetooth Recorders <deviceinfo> field:

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"1111"
ECG Type	1 = BT 3/6 (Used if nothing or illegal value is specified) 2 = BT 12
Sample Rate	1 = 100 Hz (Used if nothing or illegal value is specified) 5 = 500 Hz
Sampling Period	1 – 300 sec. (120 if nothing or illegal value is specified)
Display Mode	1 = Full Display (Used if nothing or illegal value is specified) 2 = Reduced Display
Beeper Volume	0 = Off . . 5 = Loud (Used if nothing or illegal value is specified)
Alarm Setting	Range 0 - 7 Bit 0: R-Wave detected 1 = True; 0 = False Bit 1: Electrode not connected 1 = True; 0 = False Bit 2: Heart Rate out of range 1 = True; 0 = False (0 if nothing or invalid value is specified)
Upper Heart Rate	Alarm threshold. 0 – 255. Must be higher than Lower Heart Rate threshold (120 if nothing or illegal value is specified)
Lower Heart Rate	Alarm threshold. 0 – 255. Must be lower than Upper Heart Rate threshold (40 if nothing or illegal value is specified)
Pacemaker detection	1 = Off (Used if nothing specified) 2 = On
Battery type	1 = Rechargeable batteries 2 = Alkaline batteries (Used if nothing specified)
MeasurementEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
ConnectionEvent	JavaScriptIdentifier or nothing
ConnectionCompletedEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Corscience ECG Recorder with the following settings:
device name: "TestDevice"

Bluetooth address: 00043EC204D2
Pin code: 1111
ECG Type: 1 (BT 3/6)
Sampling Rate: 1 (100 Hz)
Sampling Period: 60 (60 sec.)
Display Mode: 1 (Full Display)
Beeper Volume: 5 (Loud)
Alarm Setting: 4 (Heart Rate out of range alarm enabled)
Upper Heart Rate: 120
Lower Heart Rate: 50
Pacemaker detection: 1 (Off)
Battery type: 2 (Alkaline)
and assigning a JavaScript named “QT1” for the Measurement Event at priority 2 and another JavaScript named “QT7” for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no JavaScript will be assigned.
In this example the General Parameters are not used:

AT+pINSDV=TestDevice:CS3612ECGType:00043EC204D2-1111-1-1-60-1-5-4-120-50-1-2-QT1,2---QT7

13.25.2.3 Corscience ECG Recorder script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are “doubles”)
0	Device type	1 = BT 3/6 2 = BT 12
1	Serial number	Example: 10137
2	Battery level	0 = Critical empty 1 = Empty 2 = Okay 3 = Full
3	Pacer impulse detected	0 = No 1 = Yes
4	Heart rate limit detected	0 = No 1 = Lower limit exceeded 2 = Upper limit exceeded 3 = Both limits exceeded
5	Electrode contact	Bit 0: Electrode F Bit 1: Electrode R Bit 2: Electrode L Bit 3: Electrode N Bit 4: Electrode V1 Bit 5: Electrode V2 Bit 6: Electrode V3 Bit 7: Electrode V4 Bit 8: Electrode V5 Bit 9: Electrode V6 0 = No contact 1 = Contact

6	Protocol	4 byte value First byte (msb): Protocol version Next 2 bytes: Max. payload length Last byte (lsb): Max. number of packets that can be sent buffered by the transmitter
7	Identification	2 byte value First byte (msb): Manufacturer ID 0x01 = Corscience Last byte (lsb): Device ID 0x05 = BlueECG
8	Maintenance	4 byte value First 2 bytes: Selftest status (msb first) Last 2 bytes: Operating Cycles (Number of turn on times) (msb first)
9	Checksum error	Checksum error in any of the received packets: 0 = No 1 = Yes
10	SaveStatus	1=success, 0=No save (buffer full).
11	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
12-19	:	
20	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.25.3 Pairing with Corscience ECG Bluetooth Recorders

The RTX337x is master and the Corscience ECG is slave in the Bluetooth connection. The Bluetooth address specified in the INSDV string is used. If it is a first time connection, the Corscience ECG requests a pin code and the RTX337x responds with the pin code from the INSDV string (1111), and the ECG is afterwards locked to this RTX337x. If the Corscience ECG needs to be paired to another device, turn the ECG on and then again press the button for 20 sec. This procedure will delete the pairing.

13.25.4 Data to server

The <Monitor> part of the XML delivery format for the Corscience ECG Recorders includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: “Ecg1”.

The <DeviceId> included in messages sent from a JavaScript and auto generated message belonging to an event (e.g. MeasurementEvent) are expanded with an unique incrementing Message number. These numbers make it possible to match messages sent from the JavaScript with the belonging auto generated message.

The primary data that is delivered to the server is delivered in the format shown below *(Example)*.

Corscience ECG Bluetooth Recorders:

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<xml xmlns=”RTX-H#1-Client”>
<Gateway>
   <GatewayId>00087B00639C</GatewayId>
   <Time>1189166375</Time>
   <MessageNumber>78</MessageNumber>
   <DvType> CS3612ECGType </DvType>
   <GwInfo>…</GwInfo>
    <GwStatus>0</GwStatus>
   <GwChksum>590941100</GwChksum>
</Gateway>
<Monitor>
   <DeviceId>07915:78</DeviceId>
    <Type>NORMAL</Type>
    <BatteryStatus>3</BatteryStatus>
   <UtcTime>1189165255</UtcTime>
    <Value>
                 <SubDev>Ecg1</SubDev>
                 <SubValue>123 1234567890</SubValue>
                 </Value>
   <DvChksum>590941122</DvChksum>
</Monitor>
</xml>
```

The primary data telegram can be accompanied by one or more secondary data telegram, which are generated by one of the event activated JavaScripts, see section 10.5.18, ClearResp(), section 10.5.19, AddToResp() and section 10.5.20, SendRespToServer). The secondary data telegrams have the exact same identification as the primary data telegrams. The type (<Type>) is USERRESPONSE.

User Response:

```
<xml xmlns="RTX-H#1-Client">
   <Gateway>
      <GatewayId>00087B00639C </GatewayId>
      <Time>.........</Time>
      <MessageNumber>85</MessageNumber>
      <DvType>Gateway</DvType>
      <GwInfo>..........</GwInfo>
      <GwStatus>....</GwStatus>
      <GwChksum>..........</GwChksum>
   </Gateway>
   <Monitor>
      <DeviceId>07915:78</DeviceId>
      <Type>USERRESPONSE</Type>
      <Value>Text added by script</Value>
      <DvChksum>........</DvChksum>
   </Monitor>
</xml>
```

13.25.4.1 Value formats

The value format for Ecg1:

Format: aaa_bbbbbbbbbb

aaa Header size in bytes. If value has less than three characters, it is left padded with zeros.
bbbbbbbbbb ECG data size in bytes.
_ Space character.

Example: 123 1234567890

13.25.5 Binary data part

The binary part consists of packets received from the ECG recorder. The complete packet including start flag (0xFC) and end flag (0xFD) is added. Octet stuffing is not removed. There is a Header part and a Data part.

The Header part:

- The Protocol packet (command 0x0100).
- The Identification packet (command 0x0500)
- The Maintenance packet (command 0x0600)
- The Firmwareversion packet (command 0x0150)
- The Configure Channel Confirm (command 0x0701)
- The Configure Device Confirm (command 0x0710)
- The Medical Parameter Confirm (command 0x0916)

The Data part:
All ECG_DATA_TRANSMISSION (0x0724) data received between the Start- and Stop Transmission command is sent (command 0x0905).

13.26 Virtual device

This section describes how to enable the Virtual device within the RTX337x. The virtual device is an internal device, covering the functionality of reminders and server connection supervision.

13.26.1 Installing the Virtual device

To install a Virtual device a ***<devicename>*** must be selected. The device name is a name chosen by the operator for use during installation. For each Virtual device inserted a new device name must be selected. The Virtual device is installed using the **<devicetype>*.***

The command for installing a Virtual device is:
AT+pINSDV=<devicename>:<devicetype>

<devicetype> for the Virtual device is: **VirtType**

DV and DVS used with INSDV:
AT+pINSDV=VIRTUALDEV:VirtType:F-13.00.00:I-00.10.00
(The timing is set to be activated at 13.00 o'clock every day. The supervision is set to be repeated with an interval of 10 minutes.)

The virtual device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier below.
The JavaScripts will be activated with a set of parameters as described in the following.

13.26.1.1 Virtual Device Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
TimingEvent	<Device name>	On Timing timeout a TimingEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.

13.26.1.2 Virtual Device <deviceinfo> field

Parameter	**Value**
TimingEvent	JavaScriptIdentifier or nothing
SupervisionEvent	JavaScriptIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the JavaScripts.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Virtual Device and assigning a JavaScript named "QT1" for the Timing Event at priority 2 and another JavaScript named "QT7" for the Supervision Event.

After the reception of the command, the Timing Event is executed one time (N-1) after one second (I-00.00.01). When all repetitions has been executed the timing schedule (N-1:I-00.00.01) is replaced with I-0 (Event controlled).
The Supervision Event is executed if there has been no successful Host Server connection within the last 12 hours (I-12.00.00). In this example the General Parameters are not used.

AT+pINSDV= VIRTUALDEV:VirtType:QT1,2-QT7:N-1;I-00.00.01:I-12.00.00

13.26.1.3 Virtual Device script parameter map

TimingEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the JavaScripts.
1-8	:	
9	GeneralScriptParameter10	

13.26.1.4 How to use a Virtual device with the RTX337x

When the Virtual device is installed in the RTX337x the system is ready for use.
The Virtual device can be used to remind the patient to do things or to supervise that server connections are made. For these events AT+pDV (timing) and AT+pDVS (supervision) are used.
Below is an example how to use AT+pDV and AT+pDVS with or after installation of the Virtual device.

13.27 A&D UC-355PBT-Ci Bluetooth Scale

This section describes how to attach an A&D Bluetooth scale to the RTX337x.

13.27.1 Compatible A&D Scales

- A&D UC-355PBT-Ci Bluetooth Scale

13.27.2 Installing the A&D UC-355PBT-Ci Bluetooth Scale

To install a A&D UC-355PBT-Ci Bluetooth Scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The A&D UC-355PBT-Ci Scale is installed using the **<devicetype>**, the **<Bluetooth address>**, the Bluetooth **<Pin code>** and parameter settings.

The command for installing a A&D UC-355PBT-Ci device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the A&D UC-355PBT-Ci Bluetooth Scale is: **ADBTWS355CIType**.
<Pin code> for the A&D UC-355PBT-Ci Bluetooth Scale is 39121440

Using Bluetooth address
Below is an example on how to insert a A&D UC-355PBT-Ci Bluetooth Scale with device name: "TestDevice" and the following Bluetooth address: 123456123456. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=DEVICENAME: ADBTWS355CIType:123456123456-39121440-

Generic installation
The scale can also be inserted as generic, which opens the RTX337x for pairing with all A&D Bluetooth scales, no matter which Bluetooth addresses they have. The device is only open for pairing with all scales until the first successful connection, after this is completed the RTX337x will only respond to this scale and is no longer considered generic.

NOTE: To prevent conflicts it is only allowed to have one Bluetooth product as generic simultaneously.
NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

AT+pINSDV=DEVICENAME: ADBTWS355CIType:FFFFFFFFFFFF-39121440-

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.27.2.1 A&D UC-355PBT-Ci Bluetooth Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.27.2.2 A&D UC-355PBT-Ci Bluetooth Scale <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"39121440"
MeasurementEvent	QuestionTreeIdentifier or nothing
SupervisionEvent	QuestionTreeIdentifier or nothing
ConnectionEvent	QuestionTreeIdentifier or nothing
ConnectionCompletedEvent	QuestionTreeIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the Question Trees.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with the following Bluetooth address: 00043EC204D2. (The Bluetooth address is placed on the back of the scale.)
And assigning a Question Tree named 'QT1' for the Measurement Event at priority 2 and another Question Tree named 'QT7' for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no Question Tree will be assigned.
In this example the General Parameters are used like this:
GeneralScriptParameter1: Maximum measurement limit in kg.
GeneralScriptParameter2: Minimum measurement limit in kg.
GeneralScriptParameter3: Target weight in kg.
GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=DEVICENAME: ADBTWS355CIType:00043EC204D2-39121440-QT1,2- - -QT7–100.0–50.0–75-15

13.27.2.3 A&D UC-355PBT-Ci Bluetooth Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	0-255 (See Communication Protocol PBT Serires - version 2.3.pdf for details)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

13.27.2.4 How to use an A&D UC-355PBT-Ci Bluetooth Scale with the RTX337X

Once the specific A&D Scale has been configured in the RTX337X, step on the scale (this turns the device on and a beep can be heard). Wait for three quick beeps and the screen to update and show "step off". The measurement will be transmitted. If nothing happens please verify the Bluetooth address and PIN code are both correct. The connection may take a minute (more if there are a multiple Bluetooth devices configured in the RTX337X).

Upon the first connection of a scale to an RTX337X, multiple stored reading may be transferred. After the first connection is completed, the scale is then configured to only store one reading.

13.27.2.5 A&D UC-355PBT-Ci Bluetooth Scale measurement delivery

The <Monitor> part of the XML delivery format for the A&D Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: "Ws1".§

13.28 A&D UC-351PBT-Ci Bluetooth Scale

This section describes how to attach an A&D Bluetooth scale to the RTX337x.

13.28.1 Compatible A&D Scales

- A&D UC-351PBT-Ci Bluetooth Scale

13.28.2 Installing the A&D UC-351PBT-Ci Bluetooth Scale

To install a A&D UC-351PBT-Ci Bluetooth Scale a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The A&D UC-351PBT-Ci Scale is installed using the **<devicetype>**, the **<Bluetooth address>**, the Bluetooth **<Pin code>** and parameter settings.

The command for installing a A&D UC-351PBT-Ci device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the A&D UC-351PBT-Ci Bluetooth Scale is: **ADBTWS351CIType**.
<Pin code> for the A&D UC-351PBT-Ci Bluetooth Scale is 39121440

Using Bluetooth address
Below is an example on how to insert a A&D UC-351PBT-Ci Bluetooth Scale with device name: "TestDevice" and the following Bluetooth address: 123456123456. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice: ADBTWS351CIType:123456123456-39121440-

Generic installation
The scale can also be inserted as generic, which opens the RTX337x for pairing with all A&D Bluetooth scales, no matter which Bluetooth addresses they have. The device is only open for pairing with all scales until the first successful connection, after this is completed the RTX337x will only respond to this scale and is no longer considered generic.

NOTE: To prevent conflicts it is only allowed to have one Bluetooth product as generic simultaneously.

NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

AT+pINSDV=DEVICENAME: ADBTWS351CIType:FFFFFFFFFFFF-39121440-

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.28.2.1 A&D UC-351PBT-Ci Bluetooth Scale Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.28.2.2 A&D UC-351PBT-Ci Bluetooth Scale <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"39121440"
MeasurementEvent	QuestionTreeIdentifier or nothing
SupervisionEvent	QuestionTreeIdentifier or nothing
ConnectionEvent	QuestionTreeIdentifier or nothing
ConnectionCompletedEvent	QuestionTreeIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the Question Trees.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the scale with the following Bluetooth address: 00043EC204D2. (The Bluetooth address is placed on the back of the scale.)
And assigning a Question Tree named 'QT1' for the Measurement Event at priority 2 and another Question Tree named 'QT7' for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no Question Tree will be assigned.
In this example the General Parameters are used like this:

GeneralScriptParameter1: Maximum measurement limit in kg.
GeneralScriptParameter2: Minimum measurement limit in kg.
GeneralScriptParameter3: Target weight in kg.
GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=DEVICENAME: ADBTWS351CIType:00043EC204D2-39121440-QT1,2- - -QT7–100.0–50.0–75-15

13.28.2.3 A&D UC-351PBT-Ci Bluetooth Scale script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	Measurement	The measurement value.
1	Measurement Timestamp	Sec. since 1-Jan-1970
2	Kg/lb	0 = lb, 1=kg
3	BatteryStatus	0-255 (See Communication Protocol PBT Serires - version 2.3.pdf for details)
4	SaveStatus	1=success, 0=No save (buffer full).
5	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
6-13	:	
14	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

13.28.2.4 How to use an A&D UC-351PBT-Ci Bluetooth Scale with the RTX337X

Once the specific A&D Scale has been configured in the RTX337X, step on the scale (this turns the device on and a beep can be heard). Wait for three quick beeps and the screen to update and show "step off". The measurement will be transmitted. If nothing happens please verify the Bluetooth address and PIN code are both correct. The connection may take a minute (more if there are a multiple Bluetooth devices configured in the RTX337X).

Upon the first connection of a scale to an RTX337X, multiple stored reading may be transferred. After the first connection is completed, the scale is then configured to only store one reading.

13.28.2.5 A&D UC-351PBT-Ci Bluetooth Scale measurement delivery

The <Monitor> part of the XML delivery format for the A&D Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: “Ws1”.§

13.29 A&D UA-767PBT-Ci Bluetooth BPM

This section describes how to attach an A&D Bluetooth BPM to the RTX337x.

13.29.1 Compatible A&D BPM

- A&D UA-767PBT-Ci Bluetooth BPM

13.29.2 Installing the A&D UA-767PBT-Ci Bluetooth BPM

To install an A&D UA-767PBT-Ci Bluetooth BPM a **<devicename>** must be selected. The device name is a name chosen by the operator for use during installation. For each external device inserted a new device name must be selected.
The A&D UA-767PBT-Ci Bluetooth BPM is installed using the **<devicetype>**, the **<Bluetooth address>**, the Bluetooth **<Pin code>** and parameter settings.

The command for installing an A&D UA-767PBT-Ci device is:
AT+pINSDV=<devicename>:<devicetype>:<Bluetooth address>-<Pin code>---------

<devicetype> for the A&D UA-767PBT-Ci Bluetooth BPM is: **ADBTBP767CIType**.
<Pin code> for the A&D UA-767PBT-Ci Bluetooth BPM is 39121440

Using Bluetooth address
Below is an example on how to insert an A&D UA-351PBT-Ci Bluetooth Scale with device name: “TestDevice” and the following Bluetooth address: 123456123456. Installation using the Bluetooth address prevents unauthorized connections to take place.

AT+pINSDV=TestDevice: ADBTBP767CIType:123456123456-39121440-

Generic installation
The BPM can also be inserted as generic, which opens the RTX337x for pairing with all A&D Bluetooth scales, no matter which Bluetooth addresses they have. The device is only open for pairing with all Blood Pressure Monitors until the first successful connection, after this is completed the RTX337x will only respond to this scale and is no longer considered generic.

NOTE: To prevent conflicts it is only allowed to have one Bluetooth product as generic simultaneously.
NOTE: The Bluetooth Address FFFFFFFFFFFF is replaced with the address of the first successfully connected device.

AT+pINSDV=DEVICENAME: ADBTBP767CIType:FFFFFFFFFFFF-39121440-

Each External Device driver in the RTX337x generates specific events on occurrences of predefined situations. To each of these events it is possible to assign a specific JavaScript and a priority level. This is done by means of one of the AT commands AT+pINSDV – Insert or List Devices or AT+pUPDDV – Update Device. The <deviceinfo> field in these AT commands is used for this. The format for assigning these events is "JavaScriptName,priority". If no priority is assigned 1 will be used as a default. This will be referred to as a JavaScriptIdentifier in the following.
The JavaScripts will be activated with a set of parameters as described in the following.

13.29.2.1 A&D UA-767PBT-Ci Bluetooth BPM Events

Event	**Activator ID / Server <DeviceId>**	**Comment**
MeasurementEvent	<Bluetooth address> :<MessageNumber>	For each received measurement a MeasurementEvent is generated.
SupervisionEvent	<Device name> :<MessageNumber>	On Supervision timeout a SupervisionEvent is generated.
ConnectionEvent	<Bluetooth address>	For each Bluetooth connection a ConnectionEvent is generated.
ConnectionCompletedEvent	<Bluetooth address>	For each successful measurement download a ConnectionCompletedEvent is generated

13.29.2.2 A&D UA-767PBT-Ci Bluetooth BPM <deviceinfo> field

Parameter	**Value**
Bluetooth address	12 character hex (Capital letters only)
Pin code	"39121440"
MeasurementEvent	QuestionTreeIdentifier or nothing
SupervisionEvent	QuestionTreeIdentifier or nothing
ConnectionEvent	QuestionTreeIdentifier or nothing
ConnectionCompletedEvent	QuestionTreeIdentifier or nothing
GeneralScriptParameter1	Numeric value with possibility of '.' as separator. These parameters are for general use in the Question Trees.
:	
GeneralScriptParameter10	

Parameter separator: "-" (0x2D)

Below is an example on how to insert the Blood Pressure Monitor with the following Bluetooth address: 00043EC204D2. (The Bluetooth address is placed on the back of the scale.)
And assigning a Question Tree named 'QT1' for the Measurement Event at priority 2 and another Question Tree named 'QT7' for the Connection Completed Event at the default priority. For the Supervision Event and the Connection Event no Question Tree will be assigned.
In this example the General Parameters are used like this:

- GeneralScriptParameter1: Maximum measurement limit in kg.

- GeneralScriptParameter2: Minimum measurement limit in kg.
- GeneralScriptParameter3: Target weight in kg.
- GeneralScriptParameter4: Timeout limit of measurement.

AT+pINSDV=DEVICENAME: **ADBTBP767CIType**:00043EC204D2-39121440-QT1,2- - - QT7–100.0–50.0–75-15

13.29.2.3 A&D UA-767PBT-Ci Bluetooth BPM script parameter map

MeasurementEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	SYS	Systolic value
1	DIA	Diastolic value
2	PUL	Pulse rate per minute
3	MAP	Mean Arterial Pressure
4	Measurement timestamp	Sec. since 1-Jan-1970
5	BatteryStatus	0-255 (See Communication Protocol PBT Serires - version 2.3.pdf for details)
6	SaveStatus	1=success, 0=No save (buffer full).
7	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
8-15	:	
16	GeneralScriptParameter10	

SupervisionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

ConnectionCompletedEvent:

Index	**Parameter**	**Format** (all parameters are "doubles")
0	GeneralScriptParameter1	These parameters are for general use in the Question Trees.
1-8	:	
9	GeneralScriptParameter10	

13.29.2.4 How to use an A&D UA-767PBT-Ci Bluetooth BPM with the RTX337X

Once the specific A&D Blood Pressure Monitor has been configured in the RTX337X, apply the cuff to the upper arm as per the user guide for the Blood Pressure Monitor. Press the "START" button on the device. The cuff will inflate and deflate and the current pressure value will be shown on screen. Once cpmplete the device will show the measurement on it's screen and the measurement will be transmitted. If nothing happens please verify the Bluetooth address and PIN code are both correct. The connection may take a minute (more if there are a multiple Bluetooth devices configured in the RTX337X).

Upon the first connection of a Blood Pressure Monitor to an RTX337X, multiple stored reading may be transferred. After the first connection is completed, the BPM is then configured to only store one reading.

13.29.2.5 A&D UA-767PBT-Ci Bluetooth BPM measurement delivery

The <Monitor> part of the XML delivery format for the A&D Bluetooth Scale includes the following tags:

- <DeviceId>
- <Type>
- <UtcTime>
- <BatteryStatus>
- <Value>
- <DvChksum>

The format used for <Value> is: "Bpm1".§

14 Labelling and Markings

The labels are attached to the back of the RTX337x.
The contents of the labels are a combination of normal text, numbers and symbols.
Please refer to the table below for an explanation of the information included in the labels.

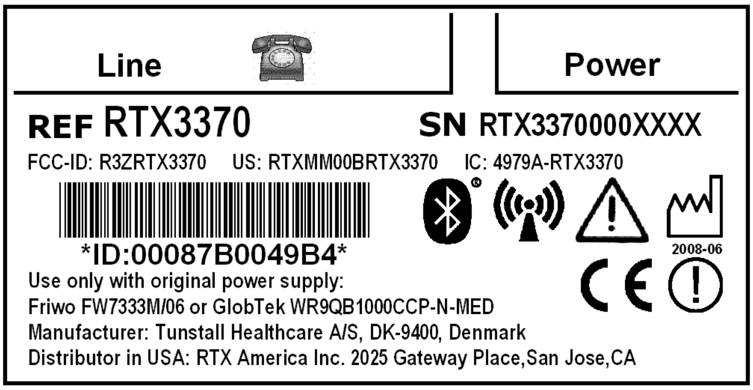
Each RTX337x unit has a unique ID, and each unit can be uniquely identified either by the BlueTooth address or by the Serial number, which are both unique.
Serial number is written in text / numbers on the label, and the BlueTooth Address is written in text / numbers and a code 39 barcode.

Information	**Description**	**Comment**
Product type	Model: "RTX3370" for the RTX3370 and "RTX3371" for the RTX3371	Product Identification
Product name	"RTX3370" for the RTX3370 and "RTX3371" for the RTX3371. In accordance with EN 980, the product name must be placed immediately after or below the " **REF** " symbol.	Product Identification & EN980
Manufacturer	Name and address of manufacturer, Tunstall Healthcare A/S, DK-9400, Denmark.	MDD 93\42\ECC & FDA requirement
Agent for distribution in US	Name and address of distributor in USA: RTX America Inc. 2025 Gateway Place, San Jose, CA"	FDA requirement
Serial number	E.g. SN: YYYYXXXXXXX Where YYYY is indicating the product and XXXXXXX is a serial number. In accordance with EN 980, the " **SN** " symbol must be placed in front of the serial number.	For production tracability & EN 980
Unique ID	E.g. ID: 1D8F709600A0	Bluetooth Address
Barcode	Info = *Unique ID* written as a barcode The barcode is a Code39 barcode.	Bluetooth Address
FCC-ID	= Grantee code + Equipment product code. E.g. 123RTX337x	Radio included

Information	Description	Comment
US-number RTX3370 only.	E.g. US: AAAEQ##TXXXXXX "= AAA (Responsible Party Code) + EQ (MM) + ##T (REN Number, received after passing Part 68 test) + XXXXXX (Product identifier = RTX337x).	PSTN included
IC-number	= Company number + Equipment product code	
Bluetooth	Symbol: ᛒ®	Bluetooth included
Read instructions before use	Symbol: ⚠	UL60601-1:2004 / IEC60601-1:2001
Contains radio transmitter	Symbol: ((•))	UL60601-1:2004 / IEC60601-1:2001
Manufacturing time	Symbol – 🏭 YYYY-MM	UL60601-1:2004 / IEC60601-1:2001
User information	Use only with original power supply: Friwo FW7333M/06 or GlobTek WR9QB1000CCP-N-MED	UL60601-1:2004 / IEC60601-1:2001
CE marking	Symbol: CE	MDD 93\42\ECC Requirement
Un-harmonised Frequency Spectrum	Symbol: ⓘ Symbol must be placed next to the CE symbol	R&TTE Requirement
Customer information to user	Area reserved for additional customer information	Optional: If required by customer
Indication of accessible connectors and buttons	Short supportive information to user about placement of accessible connectors	UL60601-1:2004 / IEC60601-1:2001

The illustrations below show as examples the labels attached to the back of the RTX3370 and RTX3371:

RTX3370 Type label:



Line ☎

Power

REF RTX3370

SN RTX3370000XXXX

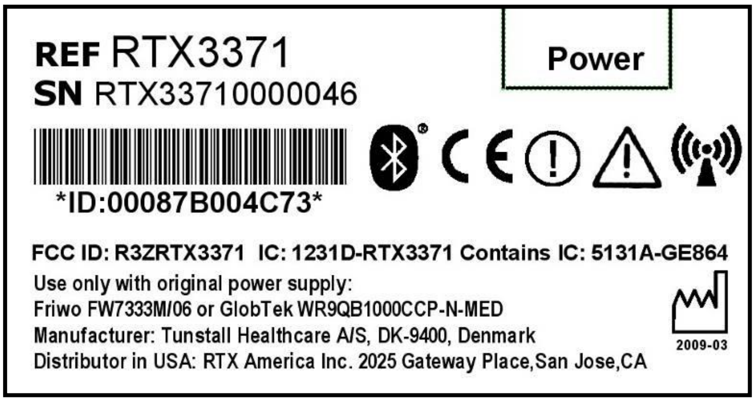
FCC-ID: R3ZRTX3370 US: RTXMM00BRTX3370 IC: 4979A-RTX3370

ID:00087B0049B4

2008-06

Use only with original power supply:
Friwo FW7333M/06 or GlobTek WR9QB1000CCP-N-MED
Manufacturer: Tunstall Healthcare A/S, DK-9400, Denmark
Distributor in USA: RTX America Inc. 2025 Gateway Place,San Jose,CA

RTX3371 Type Label:



REF RTX3371
SN RTX33710000046

Power

ID:00087B004C73

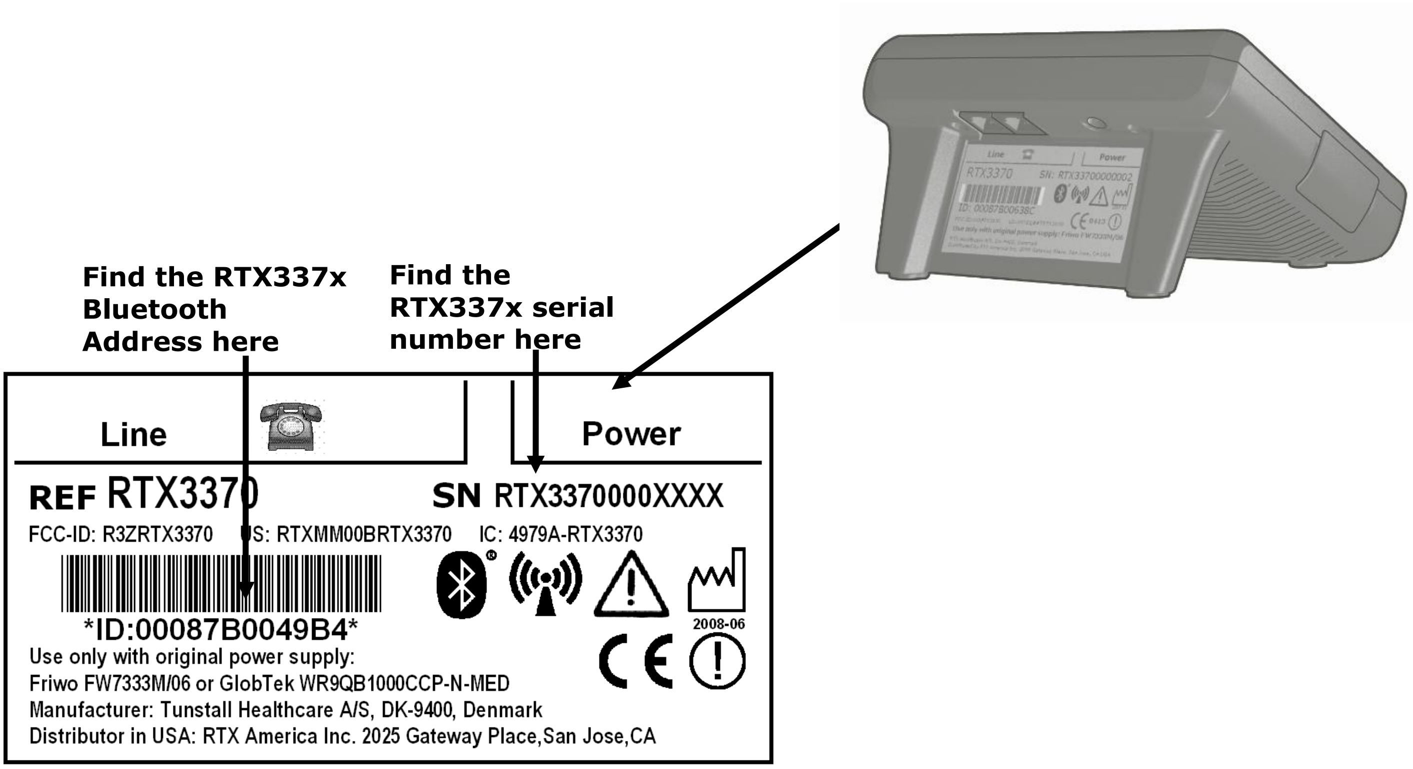
FCC ID: R3ZRTX3371 IC: 1231D-RTX3371 Contains IC: 5131A-GE864
Use only with original power supply:
Friwo FW7333M/06 or GlobTek WR9QB1000CCP-N-MED
Manufacturer: Tunstall Healthcare A/S, DK-9400, Denmark
Distributor in USA: RTX America Inc. 2025 Gateway Place,San Jose,CA

2009-03

15 Unique Identification of your RTX337x

Tunstall Healthcare service personnel have access to complete records of design changes for each product launched on the market. The information is based on the unique ID for each product.
Whenever you contact Tunstall Healthcare about your RTX337x product, please have the Unique ID available. This ensures that you obtain the most complete and accurate service information.

The Unique ID can be obtained from the label attached to the back of each RTX337x.
The Bluetooth address for the RTX337x is a unique number, which you can find as pointed out in the illustration below. The Bluetooth address can also be written out with the command 'AT+pGWID?', see section 9.1.4.
There is also a serial number on the label. This serial number also uniquely identifies your RTX337x product.



Find the RTX337x Bluetooth Address here

Find the RTX337x serial number here

Line
Power
REF RTX3370
SN RTX3370000XXXX
FCC-ID: R3ZRTX3370
US: RTXMM00BRTX3370
IC: 4979A-RTX3370
ID:00087B0049B4
2008-06
Use only with original power supply:
Friwo FW7333M/06 or GlobTek WR9QB1000CCP-N-MED
Manufacturer: Tunstall Healthcare A/S, DK-9400, Denmark
Distributor in USA: RTX America Inc. 2025 Gateway Place,San Jose,CA

16 Maintaince

For all maintenance issues of the RTX337x it is important to follow instructions from Tunstall Healthcare. Maintenance implies:

- Cleaning
- Repair
- Refurbishment
- Firmware update

These issues will be elaborated in the following sections.

16.1 Cleaning

The RTX337x may be cleaned with a soft, dry cloth or a cloth dampened with tap water and a mild detergent. A household glass cleaner on a soft, dry cloth may be used to clean off smudges. Never use any solvent, thinner, lighter fluid, alcohol, wet wipes or similar products on the RTX337x.

16.2 Repair

Repair shall be performed as specified by Tunstall Healthcare A/S:

- Only trained personnel are allowed to perform repairs.
- Instructions specified by Tunstall Healthcare A/S shall be followed and only repair and changes specified by Tunstall Healthcare A/S can be done.
- The instructions will also specify the required test procedures.
- If a repair procedure is not available for a needed repair the RTX337x must be scrapped.

All repair instructions will be reported and updated continuously to correspond the arising issues.

16.2.1 Battery for Real Time Clock

The RTX337x is equipped with an internal battery as backup power supply to the real time clock. This is to maintain a valid time and date if the main power supply is off.

The battery is pre-installed from factory and is a rechargeable type. The battery is empty when delivered and it will take about 6 hour to be fully charged. When fully charged the battery can retain the time for at least 30 days.

Under normal operating conditions, the battery does not need to be replaced throughout the lifetime of the device. If the battery needs to be replaced, please follow normal repair instructions.

The battery is located on the PCB next to the symbol: ⚠ and the accompanying text "Lithium battery". For type of battery, please see section 17, RTX337x Technical Specifications, Real time clock backup battery.

16.3 Refurbishment

Before the RTX337x can be passed from one patient to another, it has to undergo a refurbishment procedure.

16.4 Firmware update

Updating the firmware in a RTX337x must be handled like a repair, and shall be performed as specified by Tunstall Healthcare A/S:

- Only trained personnel are allowed to perform firmware update.
- Instructions specified by Tunstall Healthcare A/S shall be followed.

See section 16.4.1 for update using cable and section 16.4.2 for remote update.
A test sheet is printed showing the test parameters and the new firmware version. The Device Master Record (DMR) version is not changed.
Test sheet is signed by the operator and filed. Information about the new firmware version must be updated in the Tunstall Healthcare RTX337x database. This could be done by using a Tunstall Healthcare A/S web-interface, file exchange or direct database connection.

16.4.1 Firmware update using RS232 cable

Headlines from the procedure:

- Connect the RTX337x to the PC using a RS232 cable.
- Enable the RS232 configuration interface on the RTX337x if not already done.
- Start the Tunstall Healthcare test program on the PC (same program used at refurbishment).
- Information about the ID number, DMR and actual firmware version will be shown on the PC monitor.
- The operator looks up in a Tunstall Healthcare spread sheet showing the newest available firmware for the particular DMR, or the PC performs the look up automatically.
- New firmware is downloaded and new firmware version will be programmed in RTX337x.
- Normal test procedure like testing the display, buttons, speaker etc is performed similar to the test done in running production.
- Test sheet is printed showing the test parameters and the new firmware version. DMR version is not changed.
- Test sheet is signed by the operator and filed. Information about the new firmware version must be updated in the Tunstall Healthcare RTX337x database. Could be done using a Tunstall web-interface, file exchange or direct database connection.

16.4.2 Firmware update done remotely

In general firmware update should only be done remotely when it is critically needed. The reason for this is the possible errors leading to malfunction, and the long connection time.
Headlines from the procedure:

- A server application must be able to handle the firmware update.
- It is defined at the server side which RTX337x units to be updated with new firmware.

- The next time the specific RTX337x connects to the server the firmware update will be performed by transmitting a look up table from the server to the RTX337x, so that the RTX337x can decide which software versions to allow for download.
- Now the server can ask the RTX337x about whether a specific software version is allowed, and the RTX337x will accept or refuse.
- If RTX337x accepts the new firmware it can be transmitted and the new firmware version programmed in RTX337x.
- Now the server application must wait until next connection (maybe the next day) for new data from the updated RTX337x's to be sure that the intended use of the products is intact.
- This means that if the firmware update is performed successfully and normal data is received the firmware update is approved.
- Firmware update sheet is then printed listing all the updated RTX337x with information about ID number, DMR, old firmware version and new firmware version.
- The update sheet is signed by the responsible and trained operator and filed. Information about the new firmware version must be updated in the Tunstall Healthcare RTX337x database. Can possibly be done using a Tunstall Healthcare web-interface, file exchange or direct database connection.

17 RTX337x Technical Specifications

17.1 RTX3370

This device is for use at home.
Power requirements Mains: 100-240 VAC, 50-60Hz, 200 mA, Class II, No applied part. Only use the original power supply adapter with the type designation FW7333M/06 or WR9QB1000CCP-N-MED.
Real time clock backup battery: MF621F, Rechargeable Lithium battery
Power supply adapter: ⧈ Equipment protected by double insulation or reinforced insulation. CautionRisk of electrical shock. Dry location use only. ∼ Alternating current ⎓ Direct current
Range of temperatures Operating: +10°C to +40°C / +50 F to +104 F Storage: -10°C to +45°C / +14 F to +113 F Transportation: -20°C to +60°C / -4 F to +140 F
Relative humidity Operating/Storage/Transport: 30% to 75% RH non-condensing.
Barometric pressure Operating/Storage/Transport: 700-1060 hPa.
Dimensions (WxLxH): 145 x 125 x 75mm / 5.7 x 4.9 x 2.9 inches.
Length of cables Telephone cord: 3 m / 118 inches Power supply cable: 2 m / 79 inches
Weight: 295 g / 11 oz including power supply adapter.
Radio Transmitter: BlueTooth® compliance: Version 1.1. Operating frequency: 2.402 to 2.480 GHz. Output power: 1 to 100 mW / Bandwidth: <1MHz. Antenna type: Internal Operational range: Up to 100 meters (indoor) depending of surroundings.
The internal antenna used for this mobile transmitter must provide a separation distance of at least 20 cm from the body and must not be co-located or operating in conjunction with any other antenna or transmitter.
This device has been tested and found to comply with FCC Part 68. Please read the »US telephone network customer information« in this manual.
This device, including the power supply adapter, is certified to comply with IEC/UL60601-1.
This device, including the power supply adapter, is tested according to IEC60601-1-2.
Data integrity and security: The privacy of data transmission with this device is secured by data encryption and authentication (SSL), and meets authorized standards under the HIPAA.
Year and month of production of this device is printed underneath this symbol.
⚠ Attention: See instructions for use. This symbol is located both on the device label and on the device PCB. On the PCB it refers to the battery.

🗑 Only dispose the RTX3370 at an appropriate recycle facility for electronics. Ask Tunstall Healthcare A/S for more information.
ⓘ The frequency band is not harmonized. Use of wireless Bluetooth technology can be restricted in some countries. Contact local authorities for more information.
CE Tunstall Healthcare declares that this device is in conformance with the European Directive 93/42/ECC.

17.2 RTX3371

This device is for use at home.
Power requirements Mains: 100-240 VAC, 50-60Hz, 10W maximum (typical 2.4W), Class II, No applied part. Only use the original power supply adapter with the type designation FW7333M/06 or WR9QB1000CCP-N-MED.
Real time clock backup battery: MF621F, Rechargeable Lithium battery
Power supply adapter: ▣ Equipment protected by double insulation or reinforced insulation. Caution Risk of electrical shock. Dry location use only. ∿ Alternating current ⎓ Direct current
Range of temperatures Operating: +10°C to +40°C / +50 F to +104 F Storage: -10°C to +45°C / +14 F to +113 F Transportation: -20°C to +60°C / -4 F to +140 F
Relative humidity Operating/Storage/Transport: 30% to 75% RH non-condensing.
Barometric pressure Operating/Storage/Transport: 700-1060 hPa.
Dimensions (WxLxH): 145 x 125 x 75mm / 5.7 x 4.9 x 2.9 inches.
Length of cables Power supply cable: 2 m / 79 inches
Weight: 479 g / 17 oz including power supply adapter.
ᛒ Radio Transmitter: BlueTooth® compliance: Version 1.1. Operating frequency: 2.402 to 2.480 GHz. Output power: 1 to 100 mW / Bandwidth: <1MHz. Antenna type: Internal Operational range: Up to 100 meters (indoor) depending of surroundings.
Radio Transmitter GSM/GPRS: GSM operating frequency: GSM850: 850 MHz, GSM 900: 900 MHz, DCS1800: 1800 MHz, PCS1900: 1900 MHz GSM output power: Up to 2W GPRS class: Class 10
📡 The internal antenna used for this mobile transmitter must provide a separation distance of at least 20 cm from the body and must not be co-located or operating in conjunction with any other antenna or transmitter.
This device, including the power supply adapter, is tested according to IEC60601-1-2.
This device, including the power supply adapter, is certified to comply with IEC/UL60601-1.

Data integrity and security: The privacy of data transmission with this device is secured by data encryption and authentication (SSL), and meets authorized standards under the HIPAA.	
🏭	Year and month of production of this device is printed underneath this symbol.
⚠	Attention: See instructions for use. This symbol is located both on the device label and on the device PCB. On the PCB it refers to the battery.
🗑	Only dispose the RTX3371 at an appropriate recycle facility for electronics. Ask Tunstall Healthcare A/S for more information.
(!)	The frequency band is not harmonized. Use of wireless Bluetooth technology can be restricted in some countries. Contact local authorities for more information.
CE Tunstall Healthcare declares that this device is in conformance with the European Directive 93/42/ECC and 99/05/EC..	

17.3 Electromagnetic emission

17.3.1 IEC 60601-1-2:2001 table 201

Emissions test	Compliance	Electromagnetic environment - guidance
This device is intended for use in the electromagnetic environment specified below. The customer or the user of the device should assure that it is used in such an environment.		
RF emissions CISPR 11	Group 1	The device uses RF energy only for its internal function. Therefore, its RF emissions are very low and are not likely to cause any interference in nearby electronic equipment.
RF emissions CISPR 11	Class B	The device is suitable for use in all establishments, including domestic establishments and those directly connected to the public low-voltage power supply network that supplies buildings used for domestic purposes.
Harmonic emissions IEC 61000-3-2	Class B	
Voltage fluctuations/ flicker emissions IEC 61000-3-3	Complies	

17.4 Electromagnetic immunity

17.4.1 IEC 60601-1-2:2001 table 202

Immunity test	**IEC 60601 test level**	**Compliance level**	**Electromagnetic environment – guidance**
The device is intended for use in the electromagnetic environment specified below. The customer or the user of the device should assure that it is used in such an environment.			
Electrostatic discharge (ESD) IEC 61000-4-2	±6 kV contact ±8 kV air	±6 kV contact ±8 kV air	Floors should be wood, concrete or ceramic tile. If floors are covered with synthetic material, the relative humidity should be at least 30%.
Electrical fast transient/burst IEC 61000-4-4	±2 kV for power supply lines ±1 kV for input/output lines	±2 kV for power supply lines ±1 kV for input/output lines	Mains power quality should be that of a typical commercial or hospital environment.
Surge IEC 61000-4-5	±1 kV differential mode ±2 kV common mode	±1 kV differential mode ±2 kV common mode	Mains power quality should be that of a typical commercial or hospital environment.
Voltage dips, shorts interruptions and voltage variations on power supply input lines IEC 61000-4-11	<5% U_T (>95% dip in U_T) For 0,5 cycle 40% U_T (60% dip in U_T) For 5 cycle 70% U_T (30% dip in U_T) For 25 cycle <5% U_T (>95% dip in U_T) For 5 sec	(>95% dip in U_T) For 0,5 cycle 40% U_T (60% dip in U_T) For 5 cycle 70% U_T (30% dip in U_T) For 25 cycle <5% U_T (>95% dip in U_T) For 5 sec	Mains power quality should be that of a typical commercial or hospital environment. If the user of the device requires continued operation during power mains interruptions, it is recommended that the device be powered from an uninterruptible power supply or a battery.
Power frequency (50/60 Hz) Magnetic field	3 A/m	3 A/m	Power frequency magnetic fields should be at levels characteristic of a typical location in a typical commercial or hospital environment.
NOTE U_T is the a.c. mains voltage prior to application of the test level.			

17.4.2 IEC 60601-1-2:2001 table 204

Immunity test	**IEC 60601 test level**	**Compliance level**	**Electromagnetic environment – guidance**
The device is intended for use in the electromagnetic environment specified below. The customer or the user of the device should assure that it is used in such an environment.			
Conducted RF IEC 61000-4-6	3 Vrms 150 kHz to 80 MHz	3 Vrms	Portable and mobile RF communications equipment should be used no closer to any part of the Model 006, including cables, than the recommended separation distance calculated from the equation applicable to the frequency of the transmitter.
Radiated RF IEC 61000-4-3	3 V/m 80 MHz to 2,5 GHz	3 V/m	**Recommended separation distance** $d = 1,2\sqrt{P}$ $d = 1,2\sqrt{P}$ 80 MHz to 800 MHz $d = 2,3\sqrt{P}$ 800 MHz to 2,5 GHz where P is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer and d is the recommended separation distance in metres (m). Field strengths from fixed RF transmitters, as determined by an electromagnetic site survey,[a] should be less than the compliance level in each frequency range.[b] Interference may occur in the vicinity of equipment marked with the following symbol:

			((•))

NOTE 1 At 80 MHz and 800 MHz, the higher frequency range applies.

NOTE 2 These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects and people.

[a] Field strengths from fixed transmitters, such as base stations for radio (cellular/cordless) telephones and land mobile radios, amateur radio, AM and FM radio broadcast and TV broadcast cannot be predicted theoretically with accuracy. To assess the electromagnetic environment due to fixed RF transmitters, an electromagnetic site survey should be considered. If the measured field strength in the location in which the Model 006 is used exceeds the applicable RF compliance level above, the Model 006 should be observed to verify normal operation. If abnormal performance is observed, additional measures may be necessary, such as re-orienting or relocating the Model 006.

[b] Over the frequency range 150 kHz to 80 MHz, field strengths should be less than 3 V/m.

17.4.3 IEC 60601-1-2:2001 table 206

Recommended separation distances between portable and mobile RF communications equipment and the device

The device is intended for use in an electromagnetic environment in which radiated RF disturbances are controlled. The customer or the user of the device can help prevent electromagnetic interference by maintaining a minimum distance between portable and mobile RF communications equipment (transmitters) and the device as recommended below, according to the maximum output power of the communications equipment.

Rated maximum output power of transmitter W	**Separation distance according to frequency of transmitter** M		
	150 kHz to 80 MHz $d = 1{,}2\sqrt{P}$	**80 MHz to 800 MHz** $d = 1{,}2\sqrt{P}$	**800 MHz to 2,5 GHz** $d = 2{,}3\sqrt{P}$
0,01	0,12	0,12	0,23
0,1	0,38	0,38	0,73
1	1,2	1,2	2,3
10	3,8	3,8	7,3
100	12	12	23

For transmitters rated at a maximum output power not listed above, the recommended separation distance *d* in metres (m) can be estimated using the equation applicable to the frequency of the transmitter, where *P* is the maximum output power rating of the transmitter in watts (W) according to the transmitter manufacturer.

NOTE 1 At 80 MHz and 800 MHz, the separation distance for the higher frequency range applies.

NOTE 2 These guidelines may not apply in all situations. Electromagnetic propagation is affected by absorption and reflection from structures, objects and people.

18 Phone Line Requirements

The internal modem needs the telephone line quality to be without noise that could cause a malfunction. Since the modem is communicating using low frequency tones, the telephone noise should also be detectable using a normal parallel phone. If heavy noise can be heard by listening in a parallel phone, then the telephone line provider should be contacted and they should try to improve the telephone line quality.

To be able to use the build-in function "Line In Use", it is very important that the on hook voltage is more than 24 VDC. Some telephone centrals or local telephone exchange boards use low voltage telephone lines. In this case it will be necessary to switch off the "Line In Use" function to get the RTX337x to function correctly.

18.1 Regulations and country settings

Australia or New Zealand
If the RTX337x is used in Australia or New Zealand there must be an interval of minimum 5 seconds between automated dial attempts and a maximum of 15 attempts without any outside event.

If the RTX337x is used in Australia or New Zealand decadic dialing or pulse dialing must be disabled.

Canada
If the RTX337x is used in Canada it must be configured to make only two successive dial attempts, when dialling the server. This is according to Canadian regulations.

Europe
If the RTX337x is used in Europe there must be an interval of minimum 5 seconds between automated dial attempts and a maximum of 15 attempts without any outside event.

18.1.1 Modem settings and country codes

18.1.1.1 RTX3370

For operation according to national standards/regulations the configuration of the RTX3370 must configure the modem with the correct country setting. This can be done with either AT+pMBEG1=+GCI=XX (see section 9.2.4) or AT+pMBEG2=+GCI=XX (see section 9.2.6) commands (replace XX with the appropriate code).

The supported country settings are:
9 Australia
A Austria
F Belgium
16 Brazil
1B Bulgaria
20 Canada
26 China
27 Columbia
2E Czech Republic
31 Denmark
35 Ecuador
3C Finland
3D France
42 Germany
46 Greece
50 Hong Kong
51 Hungary
53 India
57 Ireland
58 Israel
59 Italy
0 Japan
61 South Korea
69 Luxembourg
6C Malaysia
73 Mexico
7B Netherlands
7E New Zealand
82 Norway
87 Paraguay

89 Philippines
8A Poland
8B Portugal
9C Singapore
9F South Africa
A0 Spain
A5 Sweden
A6 Switzerland
B8 Russia
FE Taiwan
B4 United Kingdom
B5 United States *(default)*

18.1.1.2 RTX3371

For operation according to national standards/regulations the configuration of the RTX3371 must configure the modem with the correct bands. This can be done with either AT+pMBEG1=#BND=X (see section 9.2.4) or AT+pMBEG2=#BND=X (see section 9.2.6) commands (replace X with the appropriate code).

The supported band settings are:
0 - GSM 900MHz + DCS 1800MHz
1 - GSM 900MHz + PCS 1900MHz
2 - GMS 850MHz + PCS 1800MHz
3 - GMS 850MHz + PCS 1900MHz

This setting is maintained in the modem even after power off so it also a possibility to use the AT#BND=X via the direct modem interface: AT+pMODEM (see section 9.5.4) or ATModemCommand(command, timeout) (see section 10.8.70) .

19 Trouble shooting list

See also FAQ on the eSupport site, www.tunstallhealthcare.com under Products.

19.1 Logging

The RTX337x keeps an internal log to assist trouble shooting, the types of information being logged is configurable. The types are:

- Typeless - messages with no type information.
- Errors – Messages indicating configuration errors or encountered error conditions
- Warnings – Messages indicating errors, but which are expected to occur during normal operations or warns about insecure settings.
- Informational – Messages providing information about the normal operation of the RTX337x.
- Debug – Messages providing additional information about events occurring in the system.
- Unknown – Messages which have not been assigned a level yet.

During trouble shooting it is a good idea to enable all log levels to ensure all available information is available for the trouble shooting personel. Outside trouble shooting the log level should be reduced to errors and warnings as the more verbose levels can overflow the log quickly.

In trouble shooting situations where the RTX337x Technical Funtion interface is active it is possible to have the log output redirected to this interface, as well as enabling additional output which is normally not logged. For details about enabling the specific levels please see description of AT+pLFLG in chapter 9 AT+p Command Set.

19.2 Check the basics

Repeating what was being performed when the problem occurred can often help solve problems.
A few minutes spent in performing these simple checks may eliminate time spent waiting for repair. Always perform the following checks:

• Check that the line socket has power.

• Check that the RTX337x is plugged into the proper AC power source.

• Check that the other cables and connectors are connected properly and operating correctly.

• Check the equipment settings in the procedure that was being used when the problem occurred.

• Check that the test being performed and the expected results are within the capabilities of the RTX337x.

If trouble is experienced with this equipment US: AAAEQ##TXXXXXX, for repair or warranty information, please contact RTX America, Inc., 2099 Gateway Place, Suite 310, San Jose, California 95110. If the equipment is causing harm to the telephone network, the telephone company may request that you disconnect the equipment until the problem is resolved.

19.3 Installation

User manual: Always make sure that the patient reads the User manual before installation.

User manual: Make sure that the illustrations in the User manual are lifelike. Symbols and drawings should reflect the reality.

Telephone line: It should be checked that the telephone line is correctly connected. Ensure that there is a direct connection to the telephone outlet in the wall. Extension cords should only be used if necessary. It should be checked that cords are live.

Phone off hook: Make sure that the patient does not try to install the RTX337x while the phone is in use somewhere in the house. Also make sure that the patient does not call for assistance/support on the same line while the installation (establishment of server connection) is ongoing.

19.4 Basic Use

Technical words and terms: Be careful not to use technical words and inside expressions in User manuals and when supporting the patients.

Awaiting information: Always inform the patient of what is going on when the RTX337x is in use. If the patient should await something to happen, inform him of this. Never just leave on a blank screen or a stand-by screen, the patient might start pressing buttons, leave or be uncertain if something is wrong with the system.

19.5 Service Interface

HyperTerminal:

- If HyperTerminal has problems transmitting characters (output is printed fine but it hangs when a key is pressed until disconnect is pressed), there are probably something wrong with the handshake signals. This might be caused by a cable without RTS/CTS connected or sometimes software errors on the machine HyperTerminal is running on.

Missing output and/or input:

- Please ensure the RTX337x is configured so the RS232 port is in Technical function interface mode and not device mode.
- Ensure the cable used has the right configuration, pins 2-3 and 7-8 should be crossed and pin 5 connected directly.

Dropped characters:

- Please ensure the cable used is a proper NULL modem cable, especially pins 7-8 should be crossed, not just shorted in each connector.
- Some build in Serial ports in PC class hardware does not respond to CTS signals in a timely manner, which can result in overflowing the FIFO in the RTX337x. Possible solutions is to use a transmit delay in the terminal emulator or adjust the hardware transmit FIFO level in windows to 1. This can be done through Control Panel->System->Hardware->Device Manager, Select the serial port used and select properties->Port settings->advanced, then adjust the transmit buffer to its lowest setting.

19.6 PSTN / ISP Connection

Line in use detection: The RTX3370's default behaviour when dialling is to check whether the line is in use before going off hook. However this might fail on low voltage lines and the feature might have to he disabled through using %V0 as part of the modem initialization string (AT+pMBEG2). The modem measures the line voltage which can be read using: AT:R6C (for details please see "Silicon Laboratories SI2415")

NOTE: This will cause a user, using a parallel telephone, to hear two clicks, one when the modem goes off hook and another when it goes on hook again. If this is used in combination with X3, X1 or X0 (blind dialling) the user will also hear DTMF tones from the modem since it does not wait for a dial tone.

Line intrusion detection is not working: The modem in the RTX3370 has an intrusion detection mechanism, which detects when a parallel phone is picked up during a call and disconnects the RTX3370 from the line so the other phone can dial. This detection has a couple of parameters, which can be used to adjust it in case there are problems with the default settings. The registers controlling this are U76 (sample rate as well as differential current level) and U77 (time from off hook to intrusion detection algorithm start).

NOTE: Special care should be taken if changes are made to U76 and U77 since the line intrusion detection might not work as intended causing the phone to be blocked for other purposes, e.g. emergency calls.

RTX3370 calling server without Dial Tone detection:
If the RTX3370 is unable to detect the dial tone while attempting to connect "the wait for dial tone" feature can be disabled using one of (we suggest using X3) X3, X1 or X0 as part of the modem initialization string (AT+pMBEG2).

NOTE: This causes the RTX3370 to dial without 'listening' for a dial tone. This, in combination with %V0, would cause a user of a parallel phone to hear the DTMF tones when the RTX3370 dials.

The S6 register can be used to change the time between the RTX3370 picking up the phone and the first digit being dialled.

RTX3370 calling server with a prefix:
There are two options which can be used to make the RTX3370 dial a prefix (9 used as an example), either "9W" or "9,," (AT+pMPRED2). The difference here is that the first dials 9 then waits for a dial tone the second waits for 4 seconds (using default timeout value in S8).

RTX3370 cannot call server, when voice message is waiting:
This will most likely be seen as a "NO DIALTONE" error as phone systems usually have some tone/voice message indicating that there are new messages, there are a couple of ways this can be worked around.
Adjust the time the modem listens for a dial tone when is goes off hook through the U34 register for example AT+pMBEG1=:U34,4000 raises the default timeout of 0x1B58 (7 seconds) to 16.3 seconds. For details about this register please see "Silicon Laboratories SI2415".
Either use blind dial through setting X3S6=30 (AT+pMBEG1). This waits for 30 seconds before the RTX3370 starts dialling other values might be more optimal.

RTX3370 sporadically disconnecting from the dial up server:
If there are problems with the RTX3370 suddenly dropping the modem connection after having already completed the PAP/CHAP negotiation there are 2 status registers ATI7 and ATI8 which provides information about the line status during the last connection, as well as a disconnect code. For details please see "Silicon Laboratories SI2415".

19.7 Server connection

Timeout during soap call: If the RTX337x returns "CInternetCom - no contact with server: Code=-1, Errnum=0, String=(null), Detail=(null)" it most likely timed out while it was talking to the server (no data received in 2 minutes).

Timeout in tcpconnect(): If the RTX337x returns the following error code: "CInternetCom - no contact with server: Code=21, Errnum=361, String=Connection refused, Detail=4connect failed in tcp_connect()" the RTX337x was not able to connect to the server. Please verify that the server is available and responsive using a web browser.

Time Out and retries: Any transmission failure to the Internet server is treated as a transmission failure of the ordinary message, causing the retry schedule to be activated as normally. As a consequence of this the server might receive the same ordinary message several times.

Invalid SSL Certificate Errors
If the RTX337x complains about SSL failures the items listed below might give an idea how to approach these issues.

- Verify that the certificate has been saved in the RTX337x correctly; there must not be any <CR> characters within the certificate itself only <LF> characters for the linebreaks. The RTX337x will return an error similar to: "CInternetCom - no contact with server: Code=23, Errnum=0, String=unable to get local issuer certificate, Detail=SSL certificate presented by peer cannot be verified in tcp_connect()" if the certificate cannot be loaded.
- Attempt to load the CA certificate into a web browser and type in the address of you server to see if this succeeds. If it fails it should provide more detailed feedback than the RTX337x or if there are other issues it might ask the user for confirmation (the RTX337x will just reject it).
- Is the certificate generated for the correct IP/hostname, the certificate must have the same IP or hostname as the RTX337x is configured to connect to.
- Verify that the RTX337x and the server have a time setting, which are within the period the certificate is valid.

NOTE: The RTX337x does not know about time zones so its time must be compared against the GMT time in the certificate.

19.8 Configuration

Insertion of devices: The timing schedule AT commands for a given device (AT+pDV and AT+pDVS) cannot be used in the same configuration telegram from the server as the AT+pINSDV command for the same device. Instead use the AT+pINSDV command to set the timeschedule.

Insertion of IR devices: By design, it is only possible to configure the RTX337x to use one IR device at the time by using the AT+pINSDV command. If attempted to insert a second IR device the following error message will be shown: "Devicetype GenIR1Type -

More than one device inserted.”. Instead it is possible to use several approved type and serial numbers when inserting GenIR1Type or GenIR1TSType IR devices.

Insertion of Cabled devices: By design, it is only possible to configure the RTX337x to use one Cabled device at the time by using the AT+pINSDV command. If attempted to insert a second cabled device one of the following error message will be shown:
If it is the same type as the inserted (FreeStyle used as example): “Devicetype FreeStyle - More than one device inserted.” or if it is another type: (LsBgmType used as example) “Device LsBgmType not permitted.”

Insertion of devices as Generic: Inserting devices as generic can produce some problems with BT devices. If there are more than one RTX337x within range with generic devices and the BT device looses its pairing (for some this only requires one failed connection attempt) it will be likely to repair with the wrong device. Other problems include not being able to ensure the RTX337x does not deliver data from another device of the same type, this would then have to be filtered on the server.

19.9 JavaScripts and MMI

Button presses: The JavaScript author should be aware that button presses only give rise to events when activated with the gw.ActivateButtons() function. When an activated button gets pressed all buttons are immediately deactivated, and must be activated again before they can trigger button events i.e. only one button event is emitted for each activation with gw.ActivateButtons().

Script priority: Be sure to fully understand section 10.8.87, JavaScript Scheduling. It is important to have fully analysed the possibility of simultaneous requests for execution of JavaScripts in cases where the sequence of JavaScript execution is of importance. This might be of importance for JavaScripts activated by external devices or JavaScripts activated by some timing issue or JavaScripts activated from the Internet server.

Script events: In the JavaScripts, be sure to handle all events carefully and properly. It is very easy to get out of synchronization with events otherwise.

General task: If JavaScripts read in the last measurement by using the gw.GetLastMeasurement function then this might give problems if two devices are started at the same time. If two devices are started at once (i.e. weight and BGM), the first JavaScript might not get to read the measurement before it being overwritten by the next device. If this happens, both JavaScript executions will be based on the same measurement.

RTX337x receives more than one measurement because of old measurements in the device: The Measurement JavaScript will be activated once for each measurement (also in cases where the patient has used the device more than once without having uploaded the measured data to the RTX337x in between measurements). The JavaScript can be in control of this by looking at the measurement timestamp, thus preventing superfluous questions to be asked.

19.10 Use with external devices

All devices:

- Make sure the patient uses the device supported by (and installed at) the RTX337x. A patient might have old or other devices in the home and by mistake or omission tried to use these devices with the RTX337x.

- If the RTX337x memory is filled, new data cannot be stored. The telephone connection should be checked to ensure that the RTX337x is able to send the data and thereby make room for new data.

A&D Bluetooth weight scale (NOT UC-351PBT-C1 or UC-355PBT-C1):

- "Data-port" switch for output on the back of the weight scale must be set on weight 'B', otherwise data will not be sent to the RTX337x.

A&D Bluetooth weight scale and A&D Bluetooth blood pressure monitor:

- Make sure that the devices are within the range for the Bluetooth connection to be established.

A&D Serial weight scale:

- The weight scale should be in 'A' mode.
- The scale must not be placed on the cable.

THT Serial weight scale:

- The scale must not be placed on the cable.

Roche IR devices:

- Devices must be placed in front of the RTX337x in a certain range for the IR window, otherwise data transfer will not be successful.

- If Devices are removed before data transfer is completed data transfer will not be successful. Data will be missing.

- If timestamps on the measurements are missing, the data will be re-sent for every data-upload, until a measurement with timestamp is made. The Patient should always set the time and date in the device.

LifeScan and TheraSense (cabled) devices:

- Cable plug-in should be checked. The plug can be in the device and in the RTX337x, but without connection. The plugs should be pressed until they are inserted completely.