



europaean space agency  
GNSS-1 Project Office

Date : 01/02/2002  
Ref : E-RD-SYS-E31-010  
Issue : 2 Rev. : 0  
Page : 1

**SISNET**

# User Interface Document

Félix Torán and Dr. Javier Ventura-Traveset

THIS DOCUMENT IS THE PROPERTY OF ESA  
IT CANNOT BE REPRODUCED IN ANY FORM  
OR DISCLOSED TO ANY THIRD PARTIES  
WITHOUT WRITTEN AUTHORIZATION FROM ESA



## GNSS-1 Project Office

Ref : E-RD-SYS-E31-010

Issue : 2    Rev. : 0    Date : 01/02/2002    Page : 2

### DISTRIBUTION LIST

GNSS-1 Documentation.

EGNOS P.O.: FT, LG, HS, JV, PM, MD, WE.

D/TOS/ETT: Juan Carlos de Mateo, Pedro Pablos, Michel Tossaint, Simon Johns.

D/APP: Hans-Hermann Fromm.

GISS: Giorgio Solari.

ESTB Web Site (Publications Section): <http://www.esa.int/navigation/estb>

...



**GNSS-1 Project Office**

Ref : E-RD-SYS-E31-010

Issue : 2    Rev. : 0    Date : 01/02/2002    Page : 3

**DOCUMENT CHANGE RECORD**

<b>Issue</b>	<b>Revision</b>	<b>Date</b>	<b>Change Status</b>	<b>Origin</b>
<b>0</b>	<b>0</b>	<b>15/11/01</b>	<b>Draft</b>	<b>FT</b>
<b>1</b>	<b>0</b>	<b>21/11/01</b>	<b>First release of the document</b>	<b>FT</b>
<b>2</b>	<b>0</b>	<b>01/02/02</b>	<b>Detailed information about the new SISNeT authorisation process included (major change of the DS2DC protocol)</b>	<b>FT</b>

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>6</b>
1.1 PURPOSE OF THE DOCUMENT.....	6
1.2 ORGANISATION OF THE DOCUMENT.....	6
1.3 READING GUIDELINES .....	6
1.4 MAJOR CHANGES INTRODUCED IN ISSUE 2.0.....	7
<b>2. THE SISNET PLATFORM .....</b>	<b>8</b>
2.1 INTRODUCTION.....	8
2.2 SISNET OBJECTIVES.....	8
2.3 IS SISNET USEFUL?.....	9
2.4 OVERVIEW OF THE SISNET PLATFORM .....	11
2.5 THE BASE STATION .....	13
2.6 THE DATA SERVER.....	14
2.7 THE USER EQUIPMENT .....	17
2.8 THE USER APPLICATION SOFTWARE.....	17
2.9 THE ESA SISNET USER APPLICATION SOFTWARE.....	18
2.10 ESA SISNET-BASED DEVELOPMENTS.....	19
2.11 REAL-TIME MONITORING OF THE ESTB PERFORMANCE .....	21
2.12 POSITIONING THROUGH THE INTERNET .....	22
2.13 REAL-TIME ANALYSIS OF THE ESTB MESSAGES .....	23
2.14 REAL-TIME MONITORING OF THE ESTB SIS STATUS THROUGH THE INTERNET.....	27
2.15 SUMMARY .....	28
<b>3. USER APPLICATION SOFTWARE DEVELOPMENT .....</b>	<b>29</b>
3.1 INTRODUCTION.....	29
3.2 HOW TO DEVELOP THE UAS?.....	29
3.3 HOW TO DEVELOP THE DS2DC CLIENT?.....	30
3.4 CONCLUSIONS.....	33
<b>4. TECHNICAL SPECIFICATIONS OF THE DS2DC CLIENT.....</b>	<b>34</b>
4.1 INTRODUCTION.....	34
4.2 THE DS2DC PROTOCOL .....	34
4.2.1 FUNDAMENTALS OF THE ESA DS2DC PROTOCOL.....	34
4.2.2 AVAILABLE DS2DC COMMANDS .....	36
4.3 THE AUTHENTICATION PROCESS.....	39
4.4 THE SINCA COMPRESSION ALGORITHM .....	40
4.5 SINCA DECOMPRESSION ALGORITHM .....	42
4.6 SUMMARY .....	44

<b>5. <u>PROGRAMMING TIPS AND SOURCE CODE EXAMPLES</u></b> .....	<b>45</b>
5.1 INTRODUCTION.....	45
5.2 IMPLEMENTATION OF THE CLIENT SOCKET.....	45
5.3 REQUESTING AND OBTAINING THE EGNOS MESSAGES.....	48
5.4 DISPLAYING THE TEXT MESSAGES BROADCAST BY SISNET .....	50
5.5 SUMMARY .....	52
<b>6. <u>REFERENCES</u></b> .....	<b>53</b>
<b>7. <u>LIST OF ACRONYMS</u></b> .....	<b>55</b>
<b><u>APPENDIX A. DS2DC COMMAND REFERENCE.</u></b> .....	<b>56</b>
AUTH COMMAND .....	56
*AUTH COMMAND .....	57
*ERR COMMAND .....	58
MSG COMMAND.....	59
*MSG COMMAND.....	60
*TXT COMMAND .....	61
<b><u>APPENDIX B. SISNET ERROR CODES AND ERROR MESSAGES.</u></b> .....	<b>62</b>

## 1. INTRODUCTION

### 1.1 Purpose of the Document

This document provides a description of the SISNET user interface at different abstraction levels. The main purpose is to provide all the necessary knowledge for the development of SISNET-based Software applications. The concepts and techniques exposed in this document shall be carefully applied in any SISNET development, in order to obtain SISNET - compliant software.

### 1.2 Organisation of the Document

The document is organised as follows:

- Chapter 1 explains the objectives of this User Interface Document (UID) and presents an overview of its contents.
- Chapter 2 provides a top-level view of the SISNET platform, demonstrating its utility and describing all the components of the platform. In addition, the SISNET power is demonstrated through some ongoing SISNET developments.
- Chapter 3 introduces all the high-level knowledge necessary for the development of SISNET applications.
- Chapter 4 focuses the development of the most important component of any SISNET application: the DS2DC client. The DS2DC protocol and the SINCA compression algorithm are explained in detail.
- Chapter 5 shows some source code samples, which may be useful for the development of SISNET applications. The source code is written in C++ language, and adapted to the Borland C++ Builder Development Environment.
- Chapter 6 presents some bibliography for further reading.
- Chapter 7 shows a list of the acronyms used in this document.

### 1.3 Reading Guidelines

The document is characterised by a top-down design. It starts from a top-level view of the SISNET platform, and goes down to low-level aspects. Therefore, this nature makes the intended readership flexible. Depending on the reader's background on Software Engineering, more or less Chapters must be read.

A reader with a basic background on Software Engineering (SE) should well follow Chapters 2 and 3, obtaining a clear top-level view of the SISNET technology and the general ideas about the UAS development process.

Chapter 4 takes contact with Software development, using an architectural and algorithmic point of view. That Section has interest for the Engineers who plan to develop a SISNET application, giving a clear idea of what to do, before start coding. Therefore, high-level knowledge about programming is required.

Finally, Chapter 5 presents the most technical information, providing source code to help in the development process. That Section is oriented to the Software developer, since it requires technical knowledge in Programming. It is important to note SISNET is platform-independent and language-independent. Therefore, this document (and the source code) can be used to implement SISNET applications under any platform and using any programming language.

#### **1.4 Major Changes Introduced in Issue 2.0**

This Issue of the Document introduces major changes with respect to the previous version (Issue 1, Revision 0). Those changes regard to a significant change on the operational philosophy of SISNeT, i.e. on the definition of the DS2DC protocol. In particular, the major change is the introduction of a new user authentication process before starting any dialogue with the SISNeT Data Server. The most relevant changes to the Document are concentrated in Chapter 4, and Appendices A and B.

These changes involve, in first place, the introduction of new authentication-related DS2DC commands, supporting the new process. In addition, the tools related to the DS2DC protocol (i.e. the Data Server and the ESA User Application Software) have been upgraded, adapting to those new commands and the authentication protocol. New version 2.0 of the Data Server and the ESA User Application Software are the first applications that implement those changes, forming the new Release 2A of the SISNeT suite, which is operational since early February 2002.

Note none of the DS2DC commands introduced in Issue 1 of this Document have been removed/modified: only some new commands have been added. In appearance, this seems to be a minor change, consisting only on adding new capabilities to SISNeT. However, the new mandatory authentication protocol avoids the access to the Data Server to older versions of the User Application Software, from early February 2002. In other words, users will only be able to access SISNeT using:

- The ESA User Application version 2.0 or greater, or
- An updated version of their own SISNeT-based applications, which must be adapted to the specifications of Issue 2.0 of this Document.

In a secondary plane, an upgrade has been introduced: some error codes and error messages (not existing in the previous version of the Document) have been added to the DS2DC specifications. That information can be found in Appendix B of this Document.

## **2. THE SISNET PLATFORM**

### **2.1 Introduction**

This Chapter provides a clear big-picture of the SISNET technology. In particular, the following questions are answered:

- What is the utility of SISNET?
- Is SISNET really useful?
- What is the general architecture of the SISNET platform?
- What is the architecture of the main SISNET components?
- What is the current status of SISNET?
- Which are the currently existing SISNET developments?

### **2.2 SISNET Objectives**

The main purpose of the SISNET platform is to provide worldwide access to the EGNOS System Test Bed (ESTB) [2,17] signal through the Internet. In other words, SISNET offers a novel technology for the development of applications integrating EGNOS-powered Satellite Navigation and Internet development. The main advantages of SISNET are the following:

- The EGNOS [1] signal is available, even if Geostationary (GEO) satellites are not visible;
- Any user equipment may have free access to a virtual EGNOS receiver, with the only condition of being connected to the Internet;
- SISNET requires a transfer rate of less than 1 kbps. This is appropriate for accessing SISNET from a mobile terminal (e.g. GSM).

For more information about the SISNET platform, the reading of [2-6, 16-18] is recommended.



### 2.3 Is SISNET useful?

SISNET can provide the EGNOS signal to land-mobile users in urban areas. In that situation the visibility of the GPS and GEO satellites is frequently poor. Hence, the question to answer is the following:

*Is really useful to have access to the EGNOS services under low visibility conditions?*

ESA has performed a preliminary study, in order to answer this question. The study is based on the use of the ESA ESPADA [7-9] simulator. The output of the study is a Technical Note [11], which is available under request. Assumptions used for this analysis may be considered conservative: they are based on specified URE of 6 meters for the GPS satellites [15], and also on specified residual corrections statistical errors for EGNOS. A major outcome of this study is illustrated in the bar plot of Figure 1.

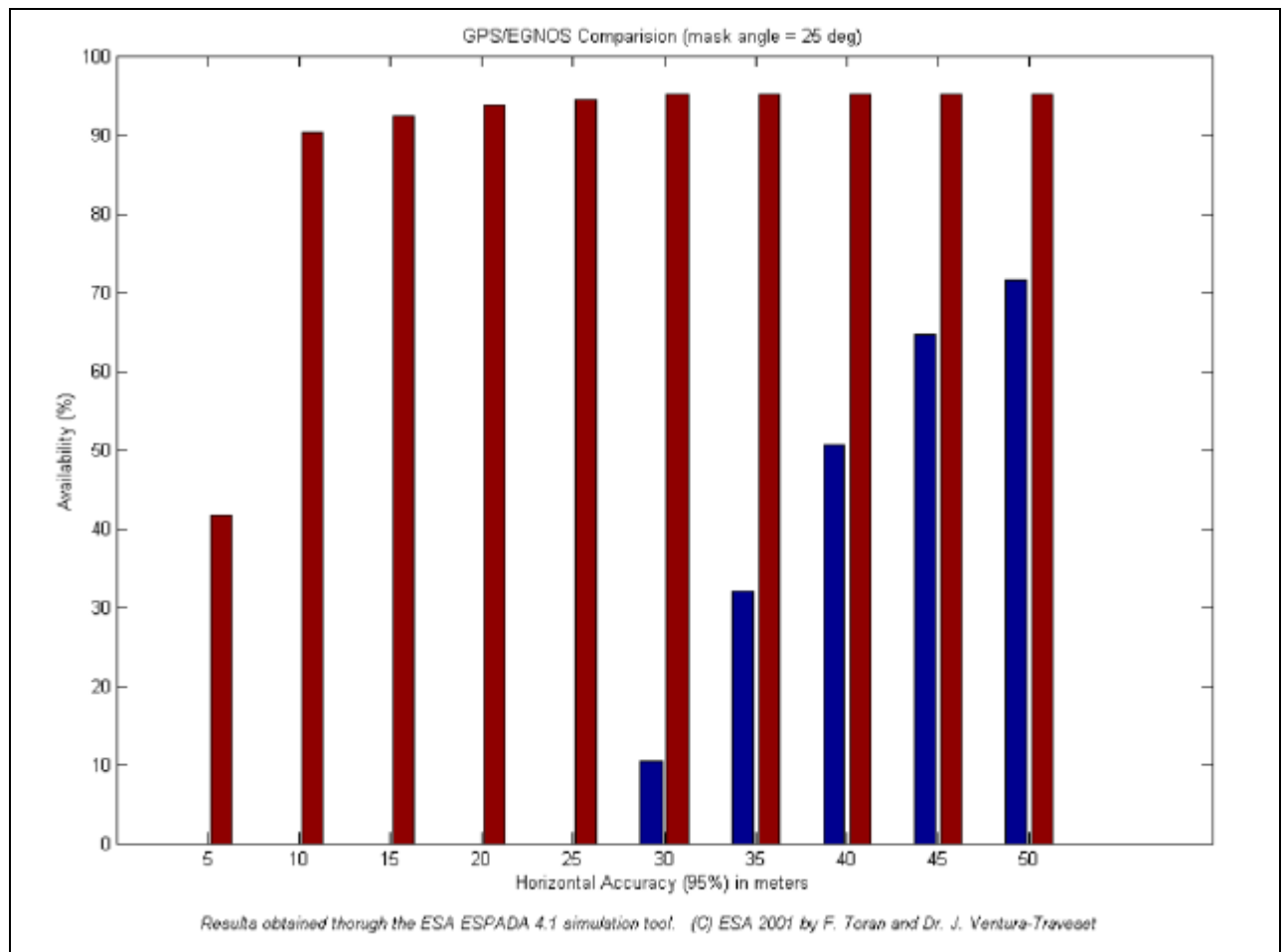


Figure 1. GPS/EGNOS comparative bar plot for a mask angle of 25 degrees.

The bar plot corresponds to a mask angle of 25 degrees (low visibility conditions). Each bar indicates the availability corresponding to an accuracy requirement (horizontal axis). The blue bars refer to a GPS-only scenario, while the red bars correspond to a GPS+EGNOS configuration.

The reading of this plot is the following:

*Under low visibility conditions (mask angle of 25 degrees):*

- *GPS offers low availability. Even in the case of 50 meters of accuracy, GPS is available less than 75% of time, while EGNOS is available 95% of the time;*
- *EGNOS is much more resistant to masking angle limitations, providing a high availability (90% to 100%) from 10 meters of accuracy.*

Taking those readings into account, a main conclusion can be formulated:

***EGNOS provides better availability of accuracy than GPS, for any masking angle. Interestingly, in situations in which user visibility of GPS satellites is limited (say to 4 - 5 satellites), the availability of the SBAS corrections plays a major role. Indeed, SBAS-available accuracy is quite insensitive to the number of GPS satellites in visibility, while GPS-only accuracy degrades significantly. SISNET benefits from this, making SBAS messages available regardless of the user visibility conditions.***

This conclusion demonstrates the SISNET potential. Even if the GEO signals are not accessible and the GPS satellites are seen with difficulty, the access through other means (e.g. GSM or GPRS via SISNET) allows taking benefit of the EGNOS power, i.e. improving the accuracy of positioning.

## 2.4 Overview of the SISNET platform

Figure 2 depicts the SISNET architecture. The four main components of the platform are the following:

- **Base Station (BS).** A PC computer connected to an EGNOS receiver through a serial port. Several software components are installed on the computer, allowing acquiring the EGNOS messages and sending it to a remote computer (the Data Server) in real-time.
- **Data Server (DS).** A high-performance computer, optimised for running server applications with a large amount of connected users. The DS functionality is implemented through a software application called SISNET Data Server (SDS). This software receives the EGNOS messages from the BS and transfers them to the remote SISNET users in real time through the Internet. In addition, the SDS implements all the extra services (present and future) provided by the SISNET system to the users.
- **User Application Software (UAS).** A Software application that accomplishes the SISNET interface specifications (exposed in this document), being able to obtain the EGNOS messages in real time (1 message per second or 250 bps) from the DS. Moreover, the UAS can access and apply the present and future additional SISNET services. Each concrete application of the SISNET platform is defined by a specific implementation of the UAS. The software can be embedded in different kind of computers and electronic devices (e.g. Personal Data Assistants).
- **Web Server.** The DS can store the received messages in a remote Web Server via the FTP protocol, enabling future development of Web / WAP applications (accessible to the users through a Web browser or mobile device).

At the moment, the BS and the DS are installed on the ESA Radionavigation Laboratory at the ESA ESTEC centre (Noordwijk, The Netherlands). The first implementation of the UAS is installed in the EGNOS Project Office (PO) in Toulouse (France), where the SISNET concept has been validated. The user-SISNET link is achieved through the ESA intranet. Figure 3 illustrates the current location of the SISNET components.

The access to SISNET is strictly limited to the ESA development team during this phase of the project, but will be available free of charge to worldwide users, after the culmination of the project.

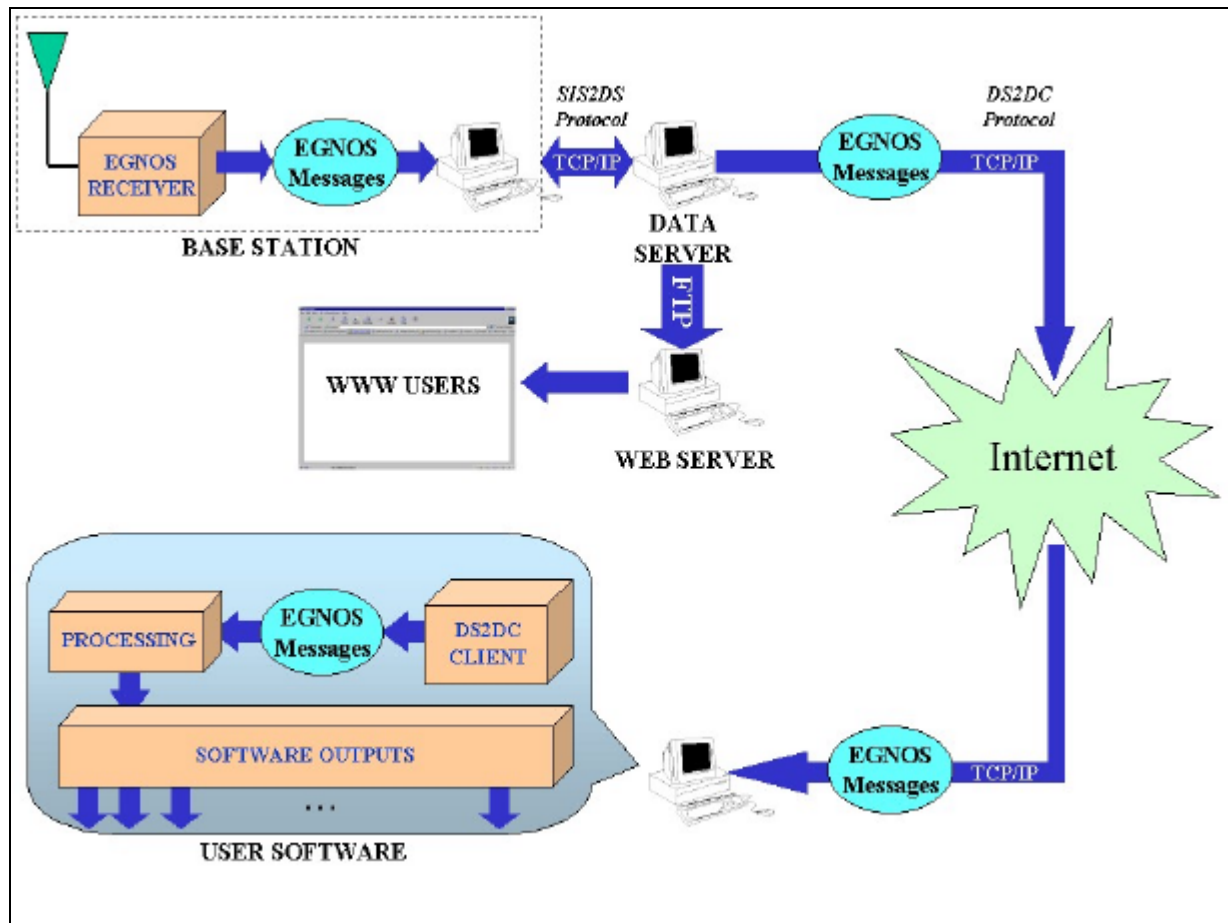


Figure 2. Architecture of the SISNET Platform.

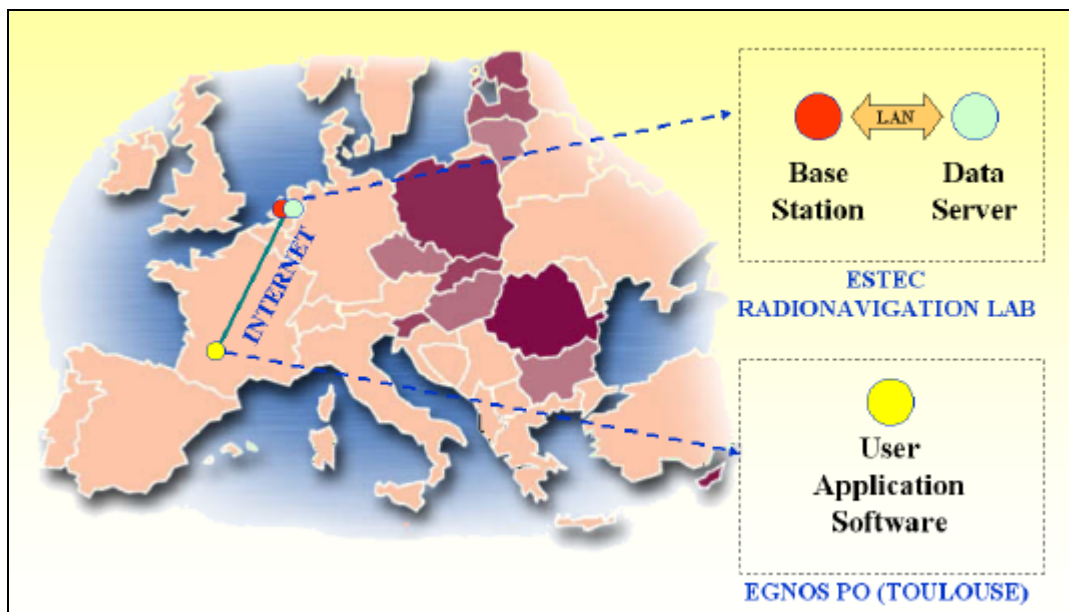


Figure 3. Current location of the SISNET components.

## 2.5 The Base Station

The BS consists of the following components (see Figure 4):

- **An EGNOS Receiver.** Currently, the BS makes use of a NovAtel Millennium receiver. The receiver is connected to a PC computer through a serial port.
- **Receiver Control Module (RCM).** A software component in contact with the EGNOS receiver. The RCM controls the receiver via the serial port, by sending standard RINEX commands. The receiver answers by sending RINEX logs to the computer each second, containing the EGNOS messages. The messages are stored on a hard disk buffer in real time (one message per second, i.e. 250 bps). The way the RCM manages the receiver is defined through an interpreted scripting language. This allows a total control of the process, and also supporting a large amount of receivers (nowadays, the most of the receivers are RINEX-compliant).
- **SIS2DS Client.** A client software component, implementing a SISNET-specific protocol called SIS2DS. The SIS2DS client takes the EGNOS messages from the buffer and sends it to the DS. SIS2DS is based on the well-known TCP/IP protocol and is optimised for transferring the EGNOS messages to the DS in real time.

The components cited above are integrated into a software application called Base Station Software (BSS), which appearance is shown in Figure 5. The main Graphical User Interface (GUI) contains the following controls:

- A. The contents of the EGNOS message extracted from the receiver and being sent to the DS. This information is updated in real time (one message per second), and presents the information in hexadecimal format. Note the presence of a 2 hexadecimal digits at the end of the message (following a '\*' symbol). Those digits act as a checksum, allowing the detection of data transmission errors.
- B. The GPS week corresponding to the current EGNOS message.
- C. The GPS time corresponding to the current EGNOS message.
- D. Log Window, containing messages about the status of the BS operation.
- E. Traffic lights, indicating the status of communications.
- F. Command buttons for the control of the BS, allowing connecting / disconnecting the BS to / from the DS, and quitting the application. These buttons are context-sensitive (they are enabled or disabled depending on the state of the BSS).

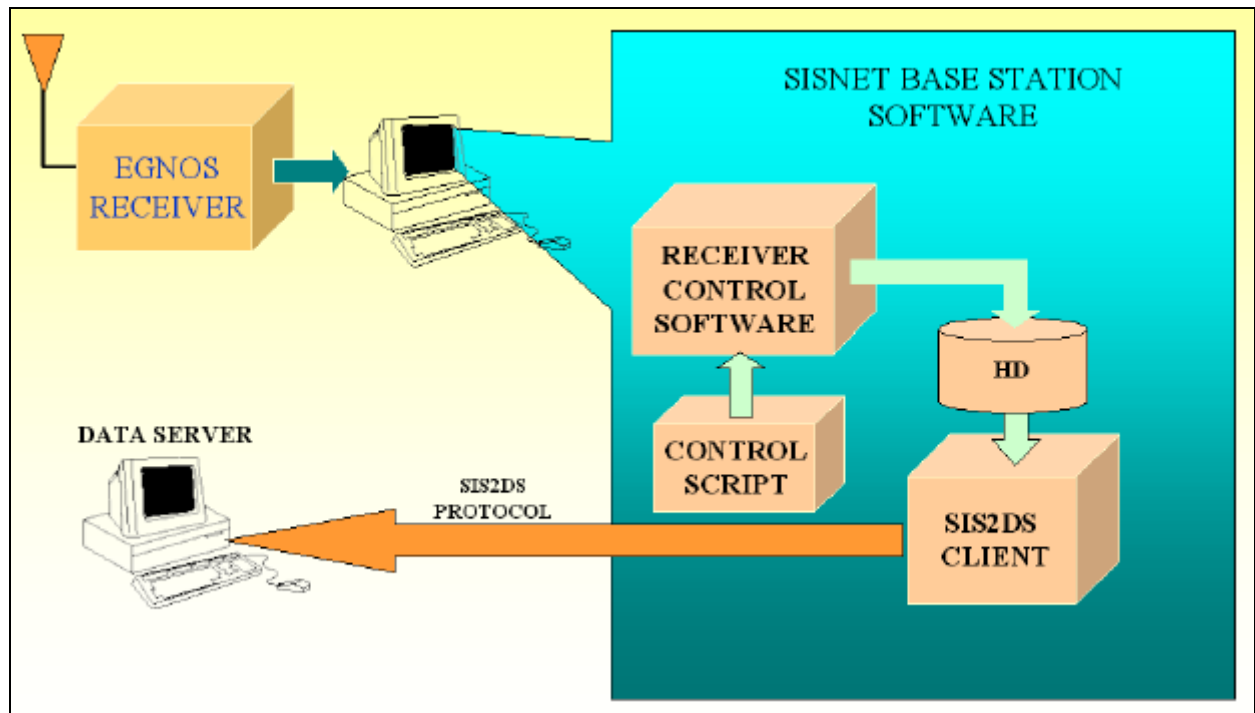


Figure 4. Architecture of the SISNET Base Station.

The BS settings window (Figure 5) requests the following parameters (to be established through agreement between BS and DS):

- G. The IP address of the DS.
- H. The TCP/IP port at which the DS listens to the BS.

The BS can run in a special way, called "simulated mode". In this mode, the EGNOS messages are taken from a file (previously logged with a receiver and the appropriate software). The simulated mode allows setting up a test platform, which is independent from SISNET. This allows performing tests without interrupting the SISNET service.

## 2.6 The Data Server

The DS just contains one component: a server application called Data Server Software (DSS). The DSS implements two server processes:

- **The SIS2DS Server**, linking the DS with the BS. This server listens to port 8888 by default.
- **The DS2DC Server**, linking the DS with the users. This server listens to port 7777 by default.

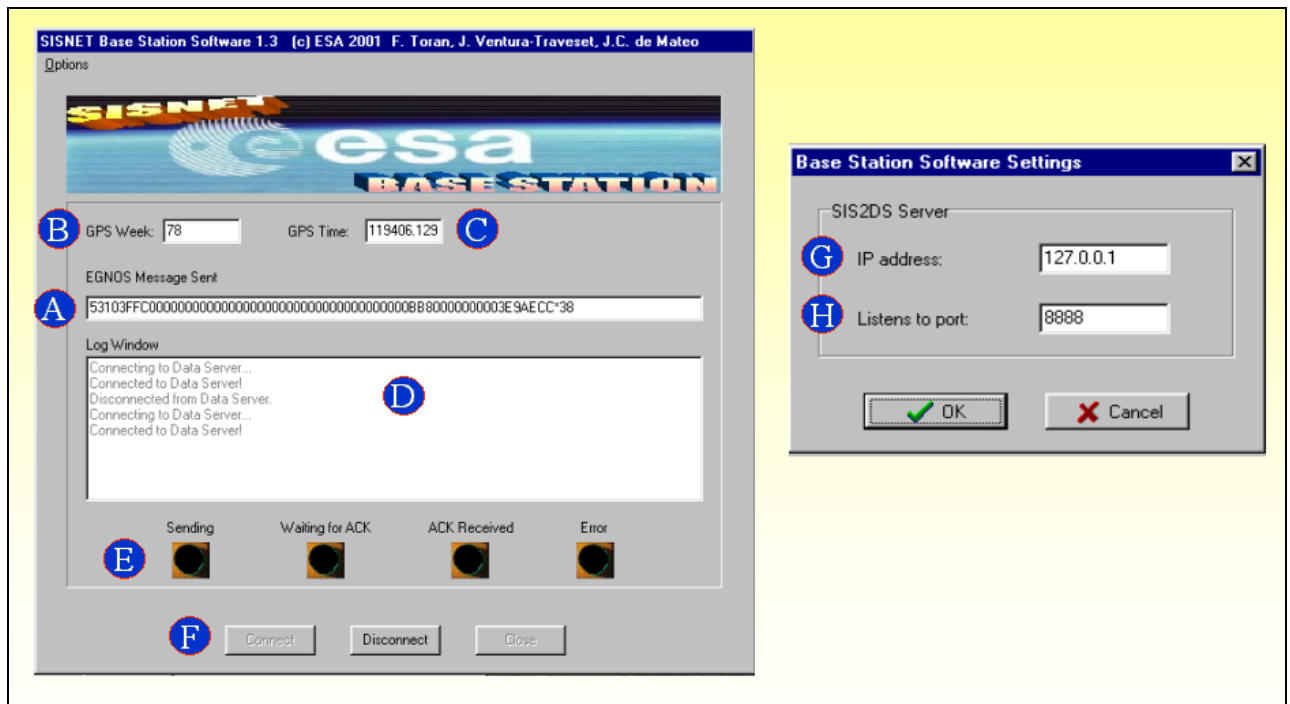


Figure 5. Base Station Software.

Figure 6 shows the appearance of the DSS. The main GUI contains the following controls:

- A. The contents of the EGNOS message received from the BS. This information is updated in real time (one message per second or even quicker, depending on the communications speed), and presents the information in the same format the BS does.
- B. The GPS week corresponding to the current EGNOS message.
- C. The GPS time corresponding to the current EGNOS message.
- D. Log Window, containing messages about the status of the DS operation. In the example shown in Figure 6, the Log Window indicates that the BS is connected to the DS and two users are also connected to the DS.
- E. Number of users currently connected to the DS.
- F. Information about the connected users (IP addresses and ports).
- G. Traffic lights, indicating the status of communications and disk storage.

The DS settings window (Figure 6) contains the following controls:

- H. The current IP address of the DS. This information is shown using a yellow background, indicating the user cannot modify it.
- I. The port at which the DS listens to the users. As stated before, the default port is 7777.

Before sending the EGNOS messages to the users, a data compression algorithm (called SINCA) is applied. The length of the EGNOS message being transferred is always less than 67 bytes. The compression algorithm frequently reduces the data size to the 25% of the original size.

The *Services* menu (Figure 6) contains only one option, allowing the broadcast of text messages to the users.

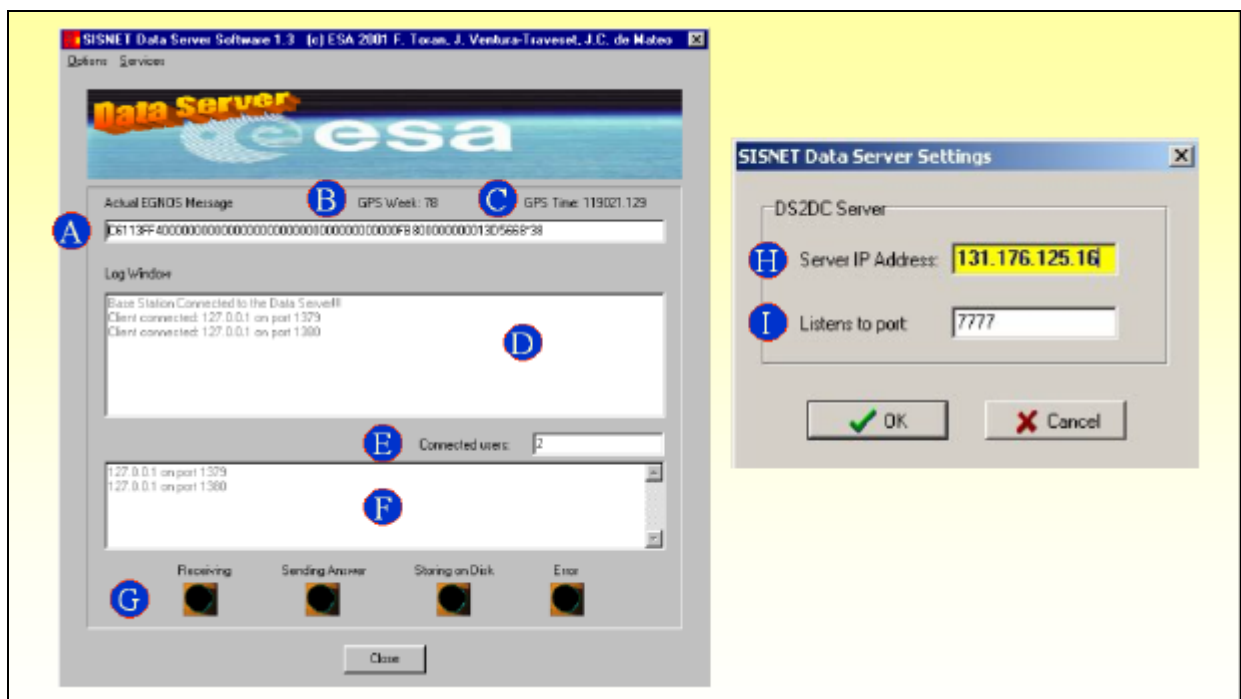


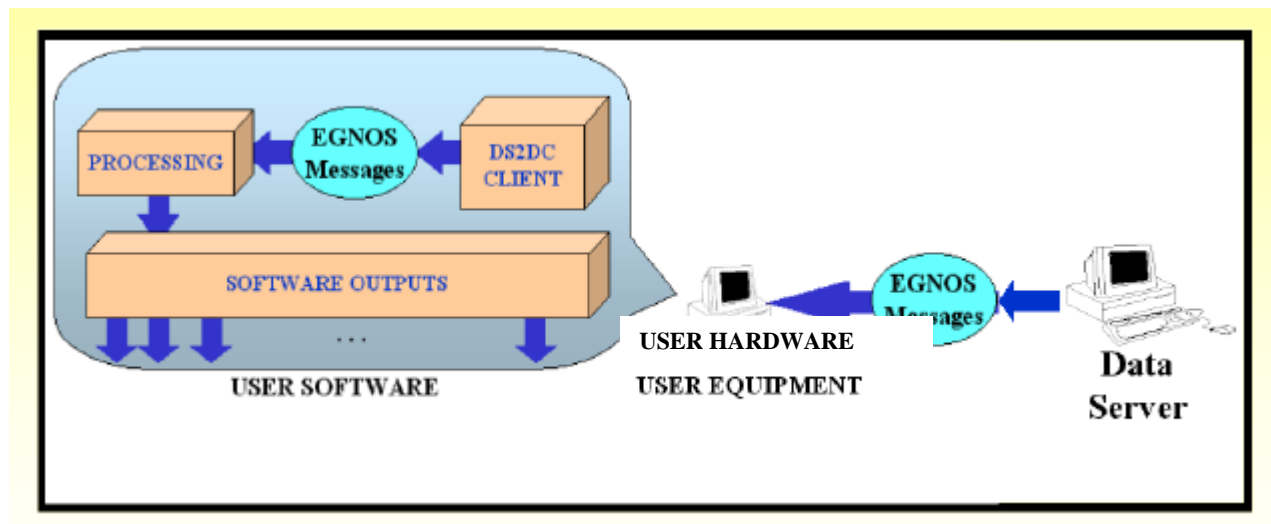
Figure 6. Data Server Software.



## 2.7 The User Equipment

The SISNET User Equipment (SUE) is the most flexible part of SISNET. Figure 7 shows the architecture of the SUE, which is made up of two main components:

- **A hardware platform**, which can be freely selected. Examples of possible hardware are PC computers and Personal Data Assistants (PDA).
- **The UAS**, running on the hardware platform. Any software developer can freely create new UAS implementations, i.e. define new SISNET-based applications.



*Figure 7. User Equipment architecture.*

## 2.8 The User Application Software

The UAS (Figure 7) includes the following software components:

- **DS2DC Client.** A client software component, implementing the DS2DC protocol. This component connects to the DS and obtains the EGNOS messages in real time. In addition, the DS2DC client can take benefit of all the additional services offered by SISNET (the number of services is continuously growing). This component shall respect all the specifications and recommendations about the DS2DC protocol (see Sections 3 to 5 of this document), in order to obtain a SISNET-compliant UAS.
- **Processing stage.** This block can be freely created by the developer, defining the functionality of each specific UAS implementation.

- **Output Interface.** The UAS developer can also freely implement this block. Generally, it consists on a GUI, adapted to the specific details of each UAS implementation.

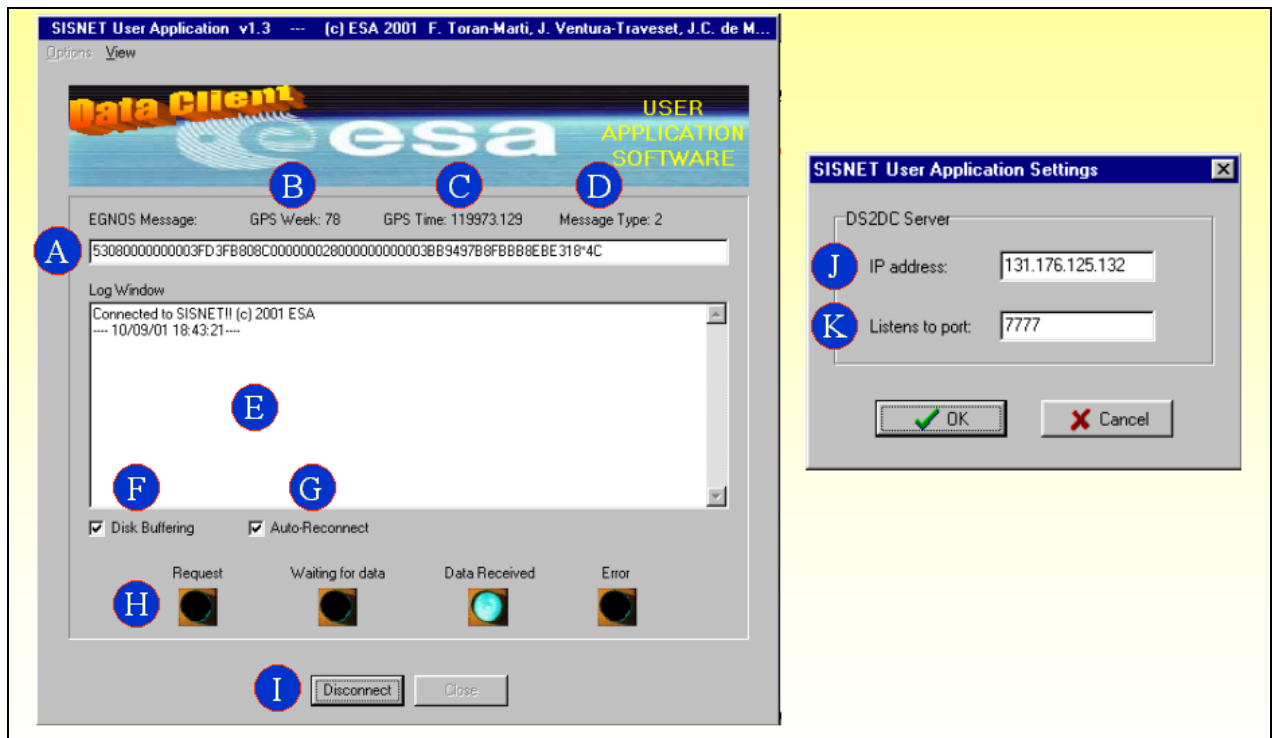
## 2.9 The ESA SISNET User Application Software

ESA has recently developed a first implementation of the SISNET UAS. The ESA UAS provides the following advantages:

- The software has been used for validating the SISNET concept;
- The UAS only presents the received EGNOS messages, i.e. acts like a monitoring tool;
- The ESA UAS is currently used as a platform for the integration of new SISNET applications. In fact, three SISNET-based applications have been developed and integrated into the UAS by ESA, in order to demonstrate the power of the SISNET technology;
- The tool will be available for free download very soon, through the ESTB Website [10].

Figure 8 shows the appearance of the ESA SISNET UAS. The main GUI contains the following controls:

- A. The contents of the EGNOS message received from the DS. This information is updated in real time (one message per second or even quicker, depending on the communications speed), and presents the information in the same format the BS and DS do.
- B. The GPS week corresponding to the current EGNOS message.
- C. The GPS time corresponding to the current EGNOS message.
- D. The message type (MT) corresponding to the current EGNOS message. The purpose and contents of each MT can be found in [12].
- E. Log Window, containing messages about the status of the UAS operation.
- F. Disk buffering option. If checked, a live buffer of EGNOS messages is maintained on the hard disk. This allows linking the ESA UAS with external applications, such as the ESA ESPADA [7-9] simulator or any software for the analysis of the EGNOS messages.
- G. Auto-reconnect option. If checked, the UAS will automatically reconnect to SISNET after any failure.
- H. Traffic lights, indicating the status of communications.
- I. Command buttons for the control of the UAS.



*Figure 8. The first ESA implementation of the UAS.*

The UAS settings window (Figure 8) contains the following controls:

- J. The IP address of the DS. This information is shown - using yellow background - in the settings panel of the DSS.
- K. The port at which the DS listens to the users. As stated before, the default port is 7777.

Note all the information shown in the UAS main GUI is updated in real time. The view menu allows launching several windows, including one devoted to the reception of text messages broadcast by the DS.

## 2.10 ESA SISNET-based developments

ESA has developed three UAS applications, demonstrating the power of the SISNET platform. Those applications have been integrated into the ESA SISNET UAS. The integration consists on substituting the processing block of the UAS by three independent processing stages, which run in parallel and define three different applications (see Figure 9). That degree of parallelism has involved an intensive use of multi-threading techniques.

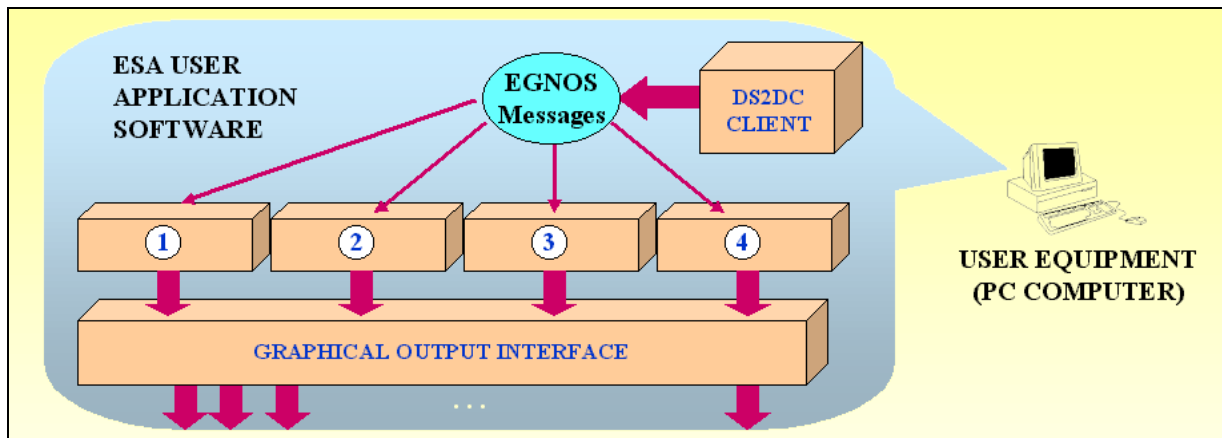


Figure 9. Integration of four parallel processing blocks into the UAS

Note Figure 9 presents four processing blocks. In fact, there is a fourth application planned to be developed by the industry.

Those applications are the following:

1. Real-time monitoring of the ESTB performance;
2. Positioning through the Internet (to be developed by the industry);
3. Real-time analysis of the ESTB messages;
4. Real-time monitoring of the ESTB Signal-In-Space (SIS) broadcast status through the Internet.

The next Sections describe with more detail those four applications.

### 2.11 Real-time monitoring of the ESTB performance

The objective of this application is to periodically provide maps of the ESTB performance. Each map must contain the Vertical Protection Limit (VPL) availability over a specific service volume, for the last 24 hours and for a given alarm limit (AL). In other words, the objective is to calculate - for each location on the map - what percentage of time VPL is under the AL.

Figure 10 depicts the architecture employed to achieve that goal. The only change with respect to the general SISNET architecture (Figure 2) is centred in the processing block of the UAS. In this case, the processing block contains an interface with the ESPADA [7-9], an advanced EGNOS simulation tool developed by ESA. The ESPADA software is able to generate offline ESTB performance maps using files, previously logged through a receiver and the appropriate software. Thanks to the new interface, ESPADA is now able to provide those maps in real time (one map each 6 minutes).

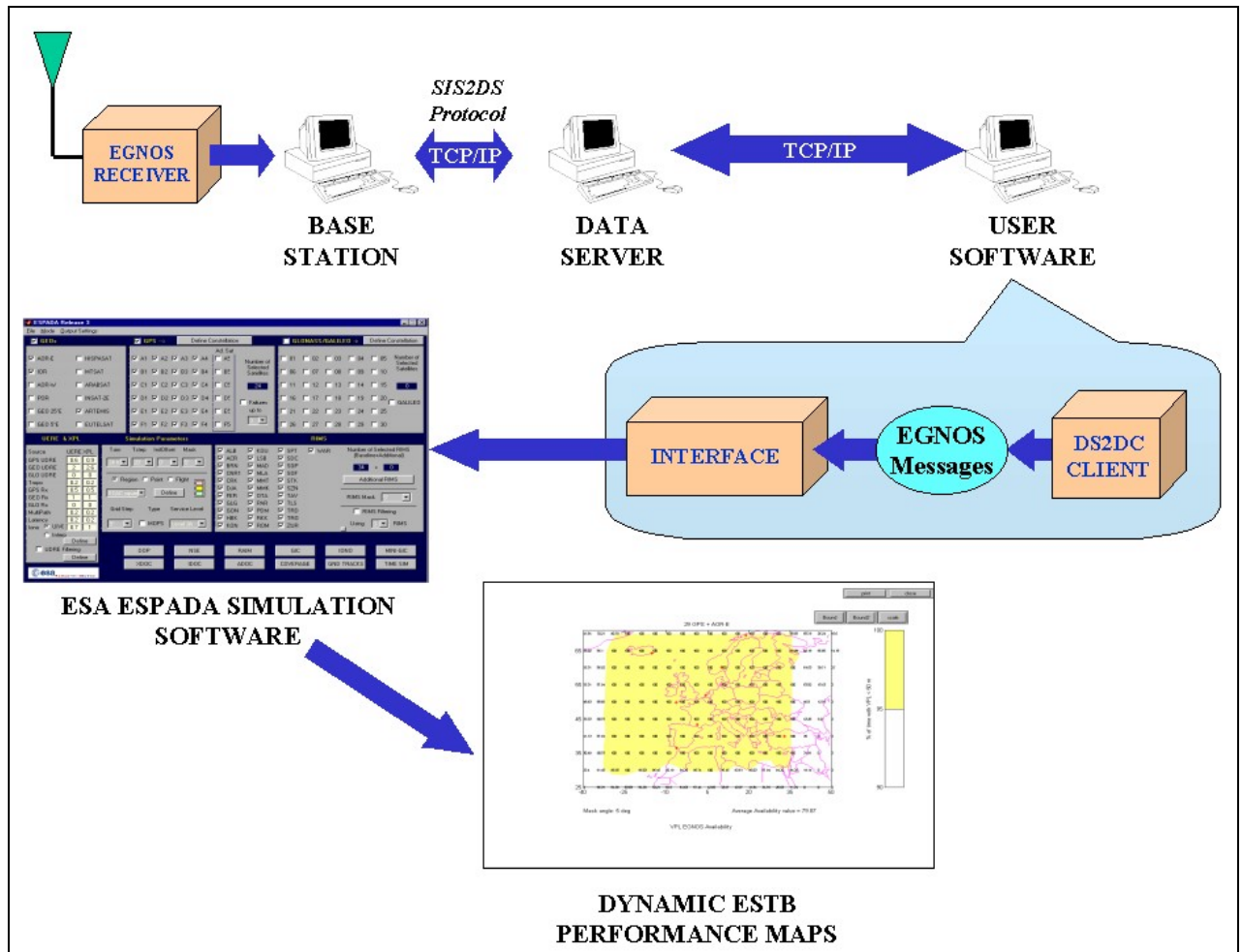


Figure 10. Real-time monitoring of the ESTB performance.

ESPADA can store the maps in two well-known formats:

- Power Point slides.
- AVI video files (generated on a daily basis).

This application has a high interest for the EGNOS monitoring activities, and significantly eases the reporting process.

### 2.12 Positioning through the Internet

The goal of this application is to provide wireless EGNOS services to land-mobile users through the Internet. Figure 11 shows the architecture of such a SISNET-based system.

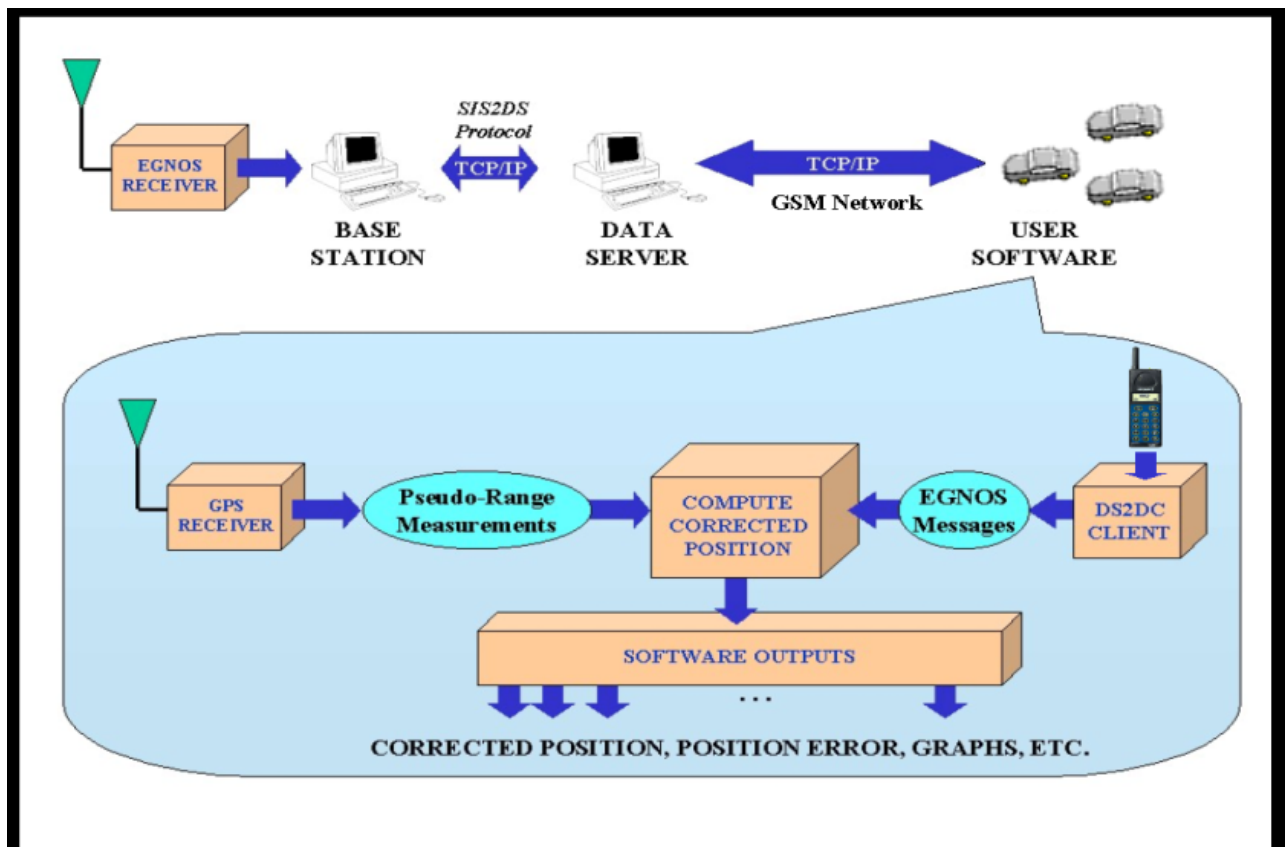


Figure 11. Positioning through the Internet.

In this case, the user equipment is embedded into cars or buses, and the connection to the Internet is achieved through a GSM or GPRS modem. The DS2DC client obtains the EGNOS messages in real-time, and a GPS receiver provides pseudorange measurements, also in real time. The processing block takes the pseudorange measurements and applies the EGNOS corrections, obtaining a significant improvement of the positioning accuracy.

This activity is planned to be developed by the industry. The results will include the development of a SISNET receiver (integrating a GPS receiver and a GSM modem), as well as demonstrations of SISNET receivers embedded into cars.

### 2.13 Real-time analysis of the ESTB messages

This application consists on the real-time analysis of the messages broadcast by EGNOS (presently, the ESTB). The objective has been achieved through the integration of several monitoring panels into the ESA UAS (Figure 12). Concretely, the "View" menu of the SISNET UAS (Figure 8) allows showing the desired panels whenever the user wants to see them. Each panel shows the contents of the EGNOS messages in a graphical and organised way, allowing the user to quickly find the desired information.

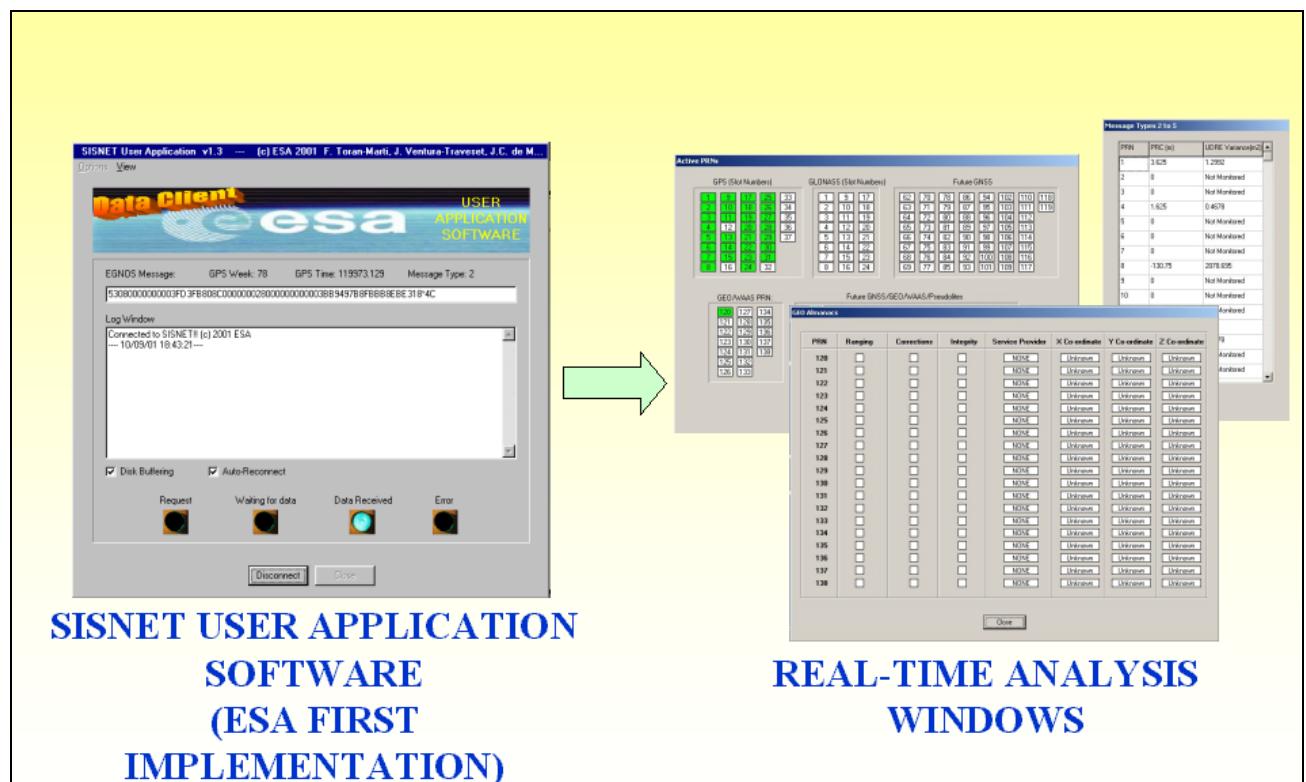


Figure 12. Real-time monitoring of the ESTB SIS status: integration of monitoring windows into the ESA SISNET UAS.

Currently, the ESA SISNET UAS integrates the following panels:

- **Analysis of MT1 (Figure 13).** This panel presents all the possible satellites that can be active, classified by constellations. The satellites are referred by the pseudorandom code number (PRN). All the active PRNs are coloured in green. Due to the specifications of MT1 [12] the maximum allowed number of active satellites is 51.
- **Analysis of MT2 to MT5 (Figure 14).** This panel shows the information regarding the fast corrections, laid out on a table. For each PRN (first column), the user can quickly find the pseudorange corrections (PRC) and the User Differential Ranging Error (UDRE) variance.
- **Analysis of MT17 (Figure 15).** This panel informs about the GEO almanacs. For each PRN corresponding to a GEO satellite, the user can know if ranging, corrections and integrity information is being broadcast. The service providers and the co-ordinates of the GEO satellites are also included.
- **Analysis of MT25 (Figure 16).** This panel shows the long-term satellite error corrections, laid out on a table. For each PRN, the table indicates the corrections to the satellites' orbits, its rates of change, the clock offset and drift, and the time of day applicability (if pertinent).

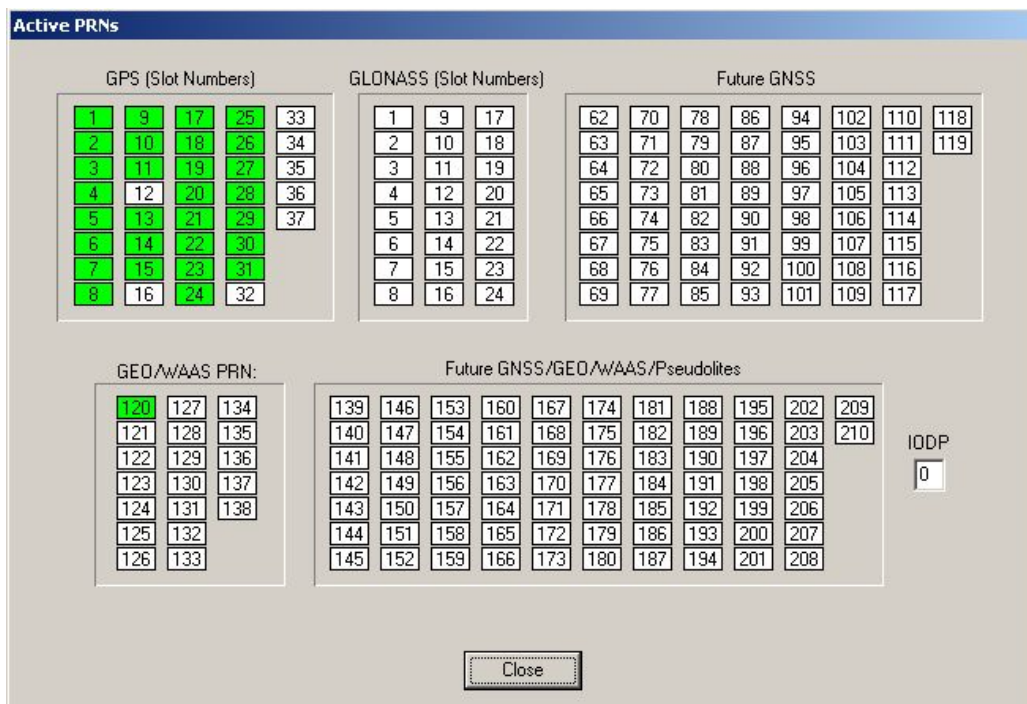
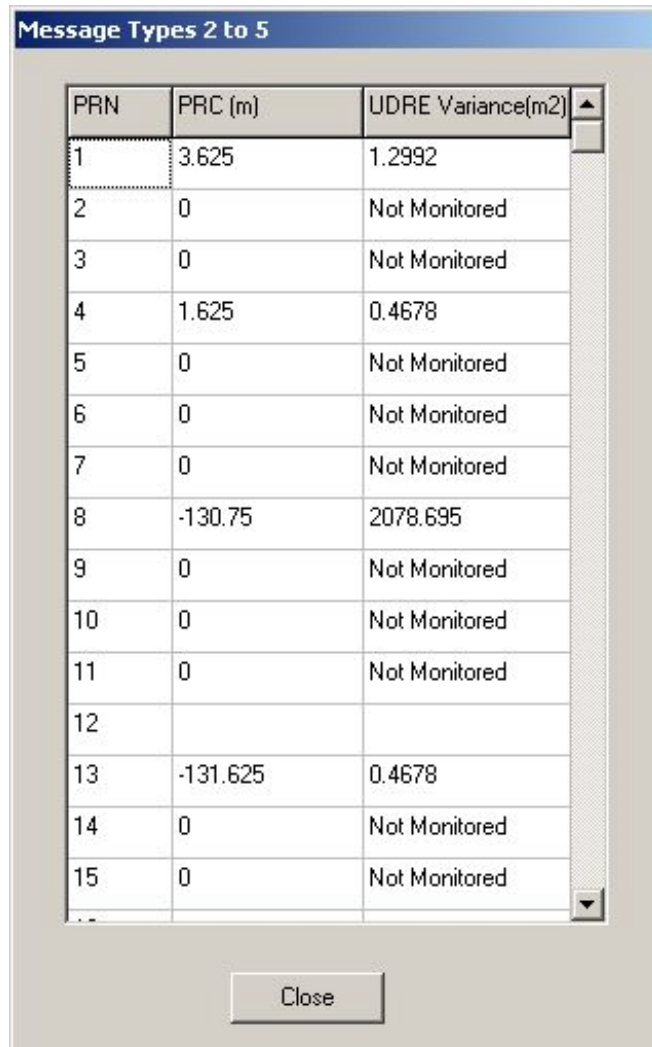


Figure 13. Real-time analysis window for MT1 (PRN mask)

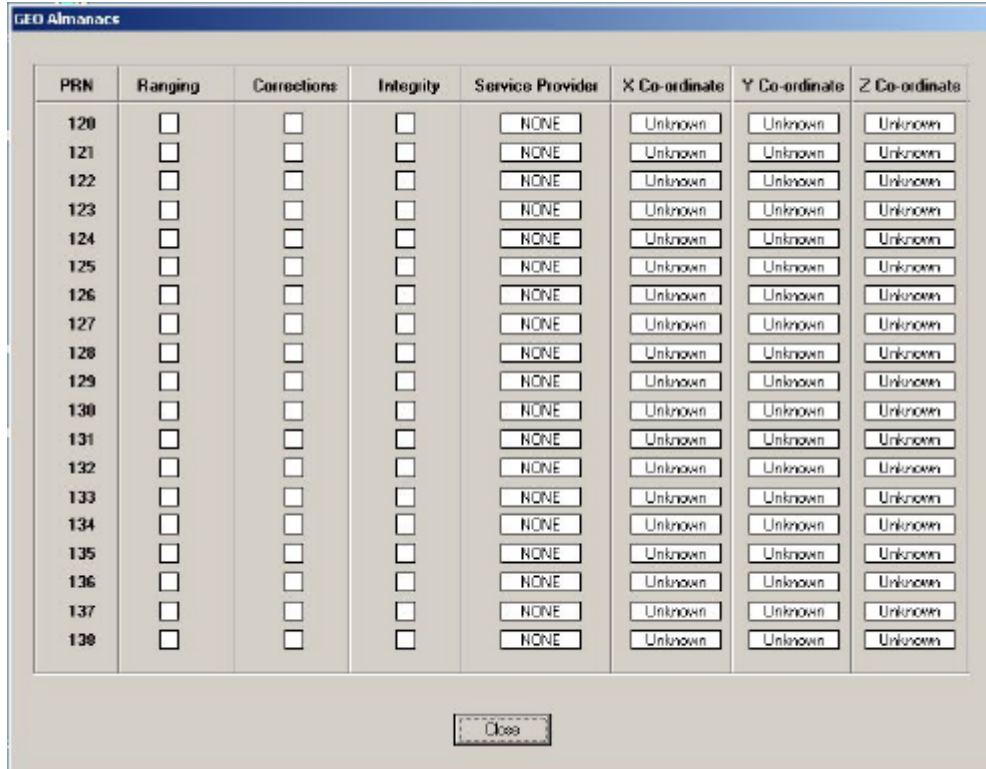


The number of monitoring panels is continuously increasing. In fact, the final goal is to provide a panel for each possible EGNOS message type.



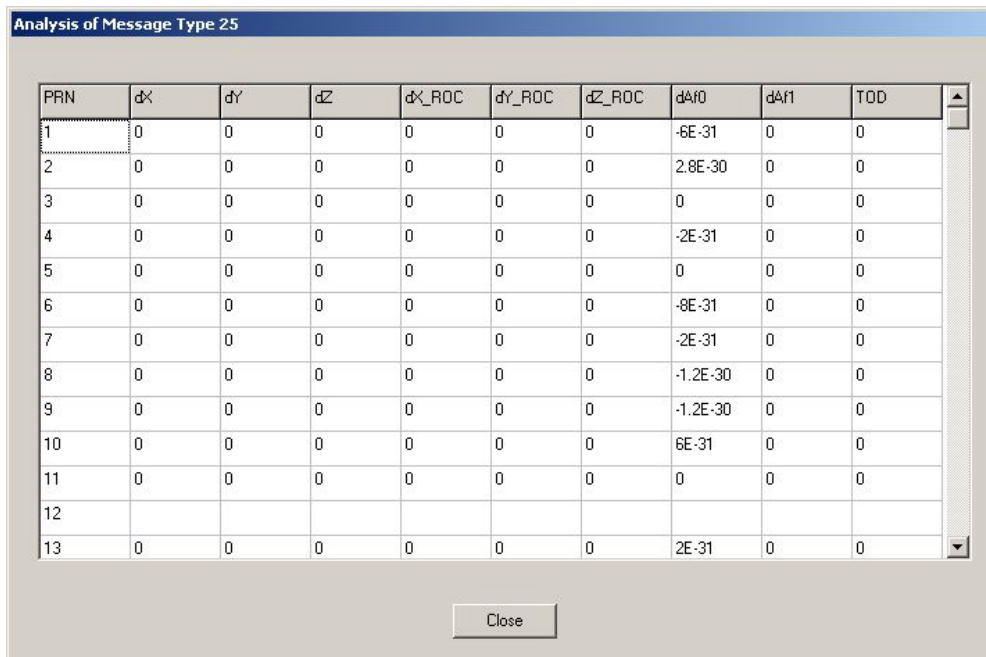
PRN	PRC (m)	UDRE Variance(m2)
1	3.625	1.2992
2	0	Not Monitored
3	0	Not Monitored
4	1.625	0.4678
5	0	Not Monitored
6	0	Not Monitored
7	0	Not Monitored
8	-130.75	2078.695
9	0	Not Monitored
10	0	Not Monitored
11	0	Not Monitored
12		
13	-131.625	0.4678
14	0	Not Monitored
15	0	Not Monitored

*Figure 14. Real-time analysis window for message types 2 to 5 (fast corrections)*



PRN	Ranging	Corrections	Integrity	Service Provider	X Co-ordinate	Y Co-ordinate	Z Co-ordinate
120	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
121	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
122	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
123	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
124	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
125	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
126	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
127	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
128	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
129	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
130	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
131	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
132	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
133	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
134	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
135	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
136	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
137	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown
138	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NONE	Unknown	Unknown	Unknown

Figure 15. Real-time analysis window for MT17 (GEO almanacs)



PRN	dX	dY	dZ	dX_ROC	dY_ROC	dZ_ROC	dA0	dA1	TOD
1	0	0	0	0	0	0	-6E-31	0	0
2	0	0	0	0	0	0	2.8E-30	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	-2E-31	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	-8E-31	0	0
7	0	0	0	0	0	0	-2E-31	0	0
8	0	0	0	0	0	0	-1.2E-30	0	0
9	0	0	0	0	0	0	-1.2E-30	0	0
10	0	0	0	0	0	0	6E-31	0	0
11	0	0	0	0	0	0	0	0	0
12									
13	0	0	0	0	0	0	2E-31	0	0

Figure 16. Real-time analysis window for MT25 (long term satellite error corrections)

## 2.14 Real-time monitoring of the ESTB SIS status through the Internet

This application allows monitoring the ESTB broadcast status and its performance in real time and through the Internet, concretely across the World Wide Web (WWW). The architecture of such a SISNET-based application is shown in Figure 17.

As introduced in Section 2.10, the ESA UAS is able to connect with the ESPADA software, generating real-time ESTB performance maps in PowerPoint format, and also as AVI video files. In addition, ESPADA is now able to generate those maps as JPEG images.

Since the ESA UAS analyses the received MTs in real time, it is possible to calculate the ESTB broadcast mode, which is fully defined by the broadcast MTs. The list of broadcast MTs corresponding to each broadcast mode can be found in [13]. The ESA UAS includes a real-time monitoring panel dedicated to the ESTB broadcast mode analysis (see Figure 18). That panel works through monitoring periods of five minutes. It indicates the current broadcast mode using an icon, and shows the received message types since the start of the current monitoring period. At the end of each monitoring interval, the icon is updated and the list of received messages is emptied. The broadcast status icon is also written to disk as a JPEG file, each time it is updated.

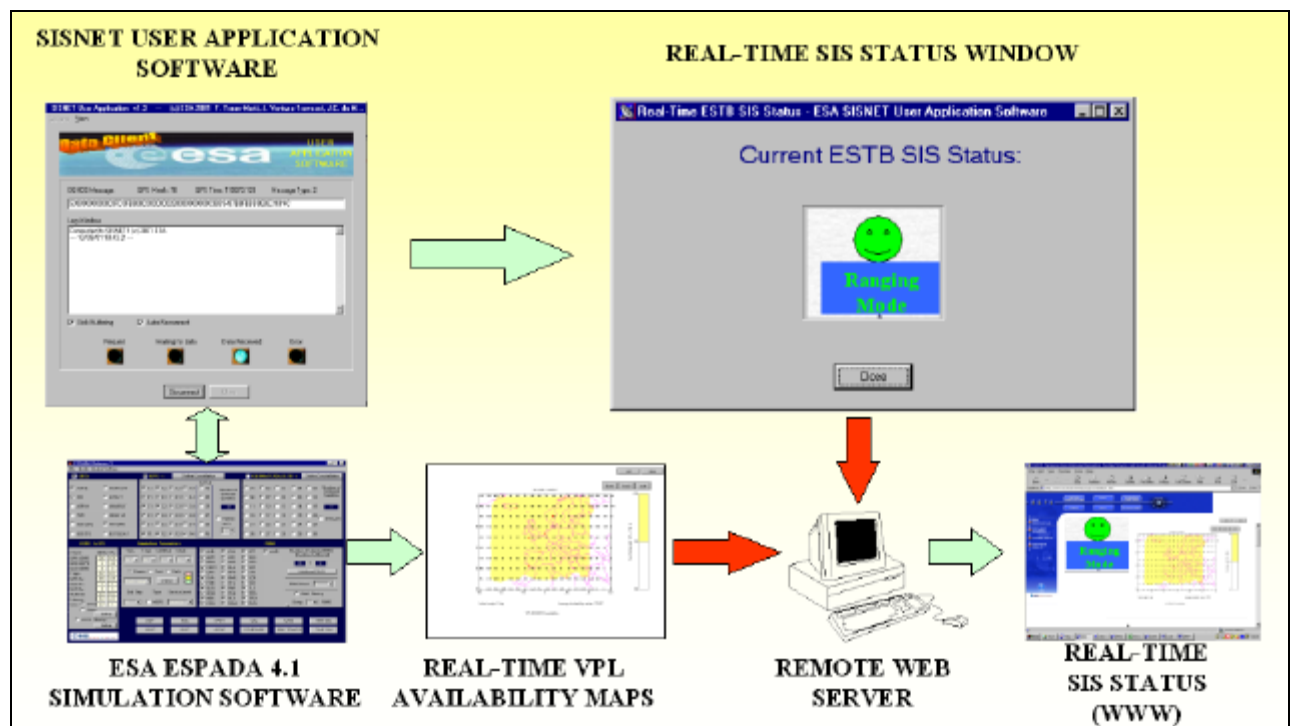


Figure 17. Web-based Real-time monitoring of the ESTB SIS broadcast status.

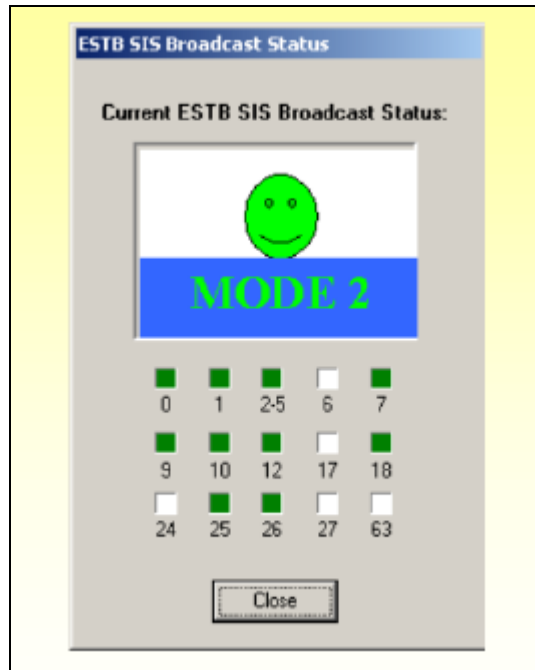


Figure 18. Real-time monitoring panel of the ESTB SIS broadcast status.

Summarising, two JPEG files are created in real-time:

- A ESPADA-produced file including a ESTB performance map;
- A UAS-produced file including the broadcast status icon.

The UAS sends both files to a remote Web Server in real time, using the FTP protocol. A web page in that server links to the JPEG files, so that the users will always see up-to-date information. The web page is auto-refreshed periodically, constituting a real-time monitoring tool for the users.

This service is now working internally at ESA, through a test platform. A pre-release version of the final system will be announced in coincidence with the 2001 ESTB Workshop event at Nice (France).

## 2.15 Summary

This Chapter has presented the top-level aspects of the SISNET technology. Firstly, the utility of the SISNET concept has been demonstrated. After that, an overview of the SISNET platform has been presented, focusing the main components and including a description of the current status of SISNET. Finally, and in order to demonstrate SISNET is a reality, several ongoing SISNET developments have been described.

Next Chapter goes to a more technical layer, showing high-level concepts regarding the development of SISNET applications.

## **3. USER APPLICATION SOFTWARE DEVELOPMENT**

### **3.1 Introduction**

The applications shown in Chapter 2 (the previous Chapter) are just an example of the power of the SISNET technology, centred in the EGNOS system monitoring and also in EGNOS-augmented positioning. Furthermore, SISNET brings Software developers the possibility of implementing their own applications, customized for any specific area of Science or Engineering. In other words, the usual SISNET developments are expected to combine:

- Any Science or Engineering field;
- The power of the Internet;
- The power of EGNOS-augmented Satellite Navigation.

This Chapter presents how to create a new SISNET application, i.e. creating a new UAS. Recommended and mandatory practices are also included. The low-level technical knowledge involved in that process is covered in Chapters 4 and 5.

The user hardware can be freely implemented and has no impact on the UAS development process. Therefore, no guidelines about hardware development are presented here.

### **3.2 How to develop the UAS?**

Once the user hardware/software platform is implemented (e.g. a PC computer with Windows 95), the application Software part of the user equipment (i.e. the UAS) must be implemented, in order to provide the desired functionality. As stated in Section 2.8, the UAS development involves the implementation of three Software components:

- The DS2DC client;
- One or more processing blocks;
- An output user interface.

The main function of the DS2DC client is to communicate with the Data Server, in order to obtain the EGNOS messages in real time. The communications protocol is called DS2DC (Data Server to Data Client), and is totally based on the well-known TCP/IP protocol, which characterizes the Internet. In addition, the DS2DC protocol allows implementing other functionalities different than the broadcast of EGNOS messages (e.g. broadcast of text messages to the users or broadcast of meteorological information based on the user's position).

The processing block works over the EGNOS messages provided by the DS2DC client, and optionally over any other external system (e.g. a GPS receiver, as explained in Section 2.11). This block defines the UAS functionality. Finally, the output interface communicates the UAS with the user, normally through a GUI.

Since the processing block and the output interface are particular to each UAS implementation, they can be freely developed without any restriction. However, the DS2DC client accomplishes a critical task: obtaining the EGNOS messages in real time. In fact, the UAS will correctly work only if the DS2DC protocol works properly. That means the design and implementation of the DS2DC client must be constrained, in order to assure the application really receives the messages broadcast by EGNOS with an appropriate performance.

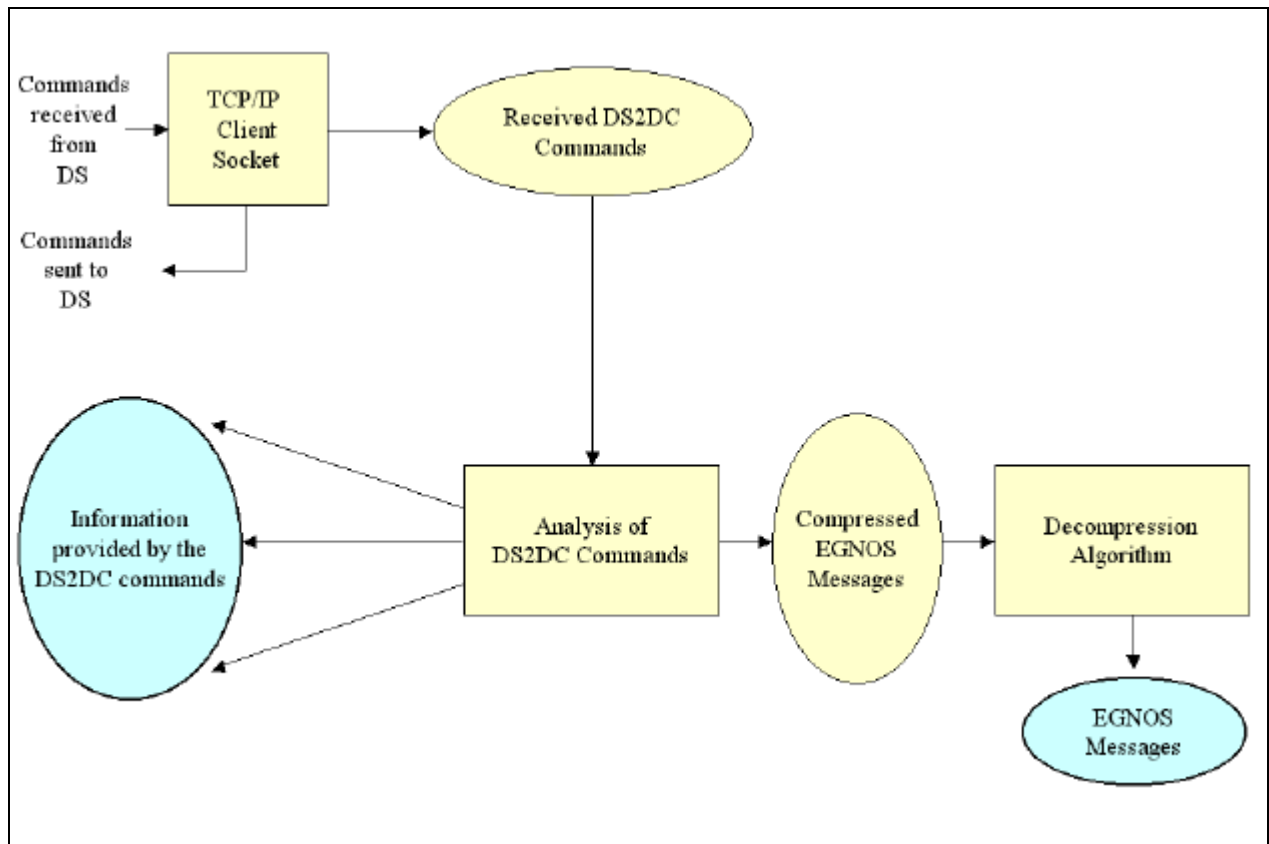
All the above can be summarised with just one sentence: the most critical part of the UAS development is the DS2DC client development. Section 3.3 (the next Section) explains how to develop such a component.

### 3.3 How to develop the DS2DC Client?

Figure 19 shows the architecture of the DS2DC client. The boxes represent functional components, while the ellipses represent information. The blue ellipses represent the output data of the DS2DC client.

The functional blocks of the DS2DC client's architecture are the following:

- **TCP/IP Client Socket (TCS).** This block opens a TCP/IP connection to the DS, using a previously agreed IP address and port. It allows sending text commands to the DS, and receiving the text commands sent by the DS as an answer (using an arbitrary port).
- **Analysis of DS2DC Commands (ADC).** This block analyses the contents of the commands sent by the DS. As a result, the information contained in those commands is extracted and converted to an easy-to-process format. In other words, this block acts like a 'signal conditioning module'.
- **Decompression Algorithm (DA).** The DS applies a compression algorithm (called SINCA) to the EGNOS messages before sending it to the users. That means a decompression algorithm must be implemented into the DS2DC client, only applicable to the commands containing EGNOS messages. The SINCA algorithm is explained in Chapter 4.



*Figure 19. Architecture of the DS2DC Client.*

The components work together as follows:

1. A TCP/IP connection to the DS is opened, using the TCS for that purpose;
2. The TCS sends the desired commands to the DS (e.g. a command requesting an EGNOS message)
3. The TCS is always listening for incoming commands from the DS. Those commands can be:
  - Answers to previously sent commands;
  - Commands spontaneously sent by the DS.
4. When a command arrives from the DS, it is captured by the TCS as a text string;
5. The received command is analysed using the ADC block. This block must extract the most relevant information contained in the message and provide it as an output of the DS2DC client. The format of that output must be as simple as possible. The goal is to ease the work to the rest of the UAS.
6. If the received command contains an EGNOS message, the DA must be applied. That block's output must be the decompressed EGNOS message (i.e. the message broadcast by the EGNOS system). The EGNOS messages are part of the DS2DC Client output.

The architecture shown in Figure 19 shall be implemented exactly as shown. The only exception is to implement the ADC block as a part of the DS2DC Client's processing block. However, this practice is not recommended, since the function of the processing block is just defining the specific functionality of the UAS, and not performing a general task like DS2DC command analysis.

The most recommended practice is to implement the DS2DC Client as a re-usable Software component (e.g. an ActiveX or VCL component). The user of the component must see the interface shown in Figure 20, in which the green ellipses indicate the input data, and the output data is represented by blue ellipses.

Using this approach, the processing block of the UAS will not receive DS2DC commands. Instead, it will receive 'pure' information:

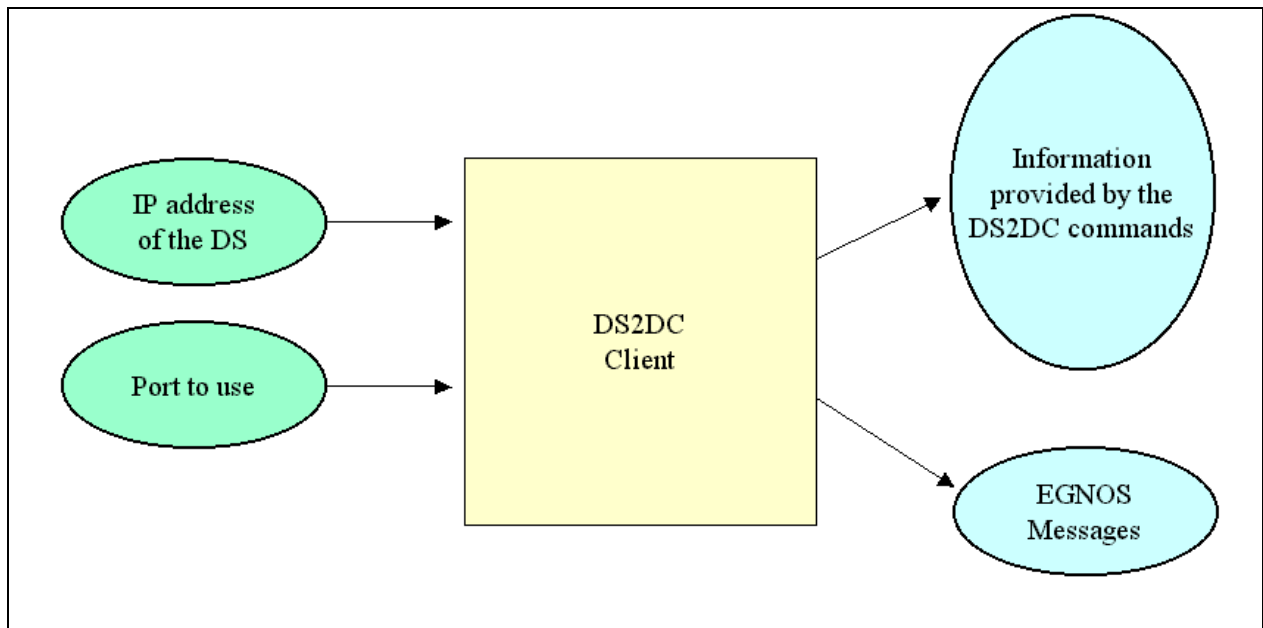
- The command type, indicating what information is provided;
- The information involved in the command, which normally answers a question invoked through a previously sent command.

All the blocks (TCS, ADC and DA) **shall** accomplish the functionality exposed in this Section. In addition, the specific requirements to respect in the development of each block are the following:

- ❑ **TCS:** must be perfectly adapted to the DS2DC protocol (explained in Chapter 4). Any non-DS2DC incoming command must be ignored and the user must be informed about that. Only DS2DC commands shall be sent to the DS.
- ❑ **ADC:** analysis must be performed strictly following the DS2DC protocol specifications.
- ❑ **DA:** the SINCA decompression algorithm must be correctly implemented.

The valid DS2DC commands are just those specified in this Document. The DS2DC Client must ignore any other command, and the user **shall** be informed about that. The developer **shall** motivate the user to inform ESA in case of receiving invalid commands (through warning windows or similar means). In addition, the DS will detect users sending invalid commands, and will apply several categories of **penalties** (including rejecting future connections). This will improve security.





*Figure 20. DS2DC Client interface.*

### 3.4 Conclusions

This Chapter has presented the process to follow when developing a new SISNET application, including recommendations and restrictions to take into account.

The main conclusions are the following:

- The development of a SISNET application is reduced to the development of the UAS, since the user hardware can be considered an independent part.
- The development of the UAS involves the creation of three blocks: the DS2DC client, a processing block and an output interface. The development is only restricted with respect to the DS2DC client.
- The development of the DS2DC client involves the creation of three blocks: the TCS, the ADC and the DA. All those blocks are strongly constrained, basically through the DS2DC protocol and SINCA algorithm specifications.

Once the process of developing the UAS is clear, some low-level details are needed, regarding the rules to respect during the development. Chapter 4 provides those technical ingredients: the DS2DC protocol and the SINCA algorithm specifications. The only way to obtain a SISNET-compliant UAS is to respect those specifications.

## 4. TECHNICAL SPECIFICATIONS OF THE DS2DC CLIENT

### 4.1 Introduction

Chapter 3 has explained the process to follow when creating a new SISNET UAS, highlighting the restrictions to take into account. Those restrictions are related to the components of the DS2DC client, and require a low-level technical explanation of:

- The DS2DC protocol specifications;
- The SINCA compression algorithm.

This Chapter provides those technical inputs, which shall be strictly respected, in order to assure the UAS is SISNET-compliant.

### 4.2 The DS2DC protocol

#### 4.2.1 Fundamentals of the ESA DS2DC protocol

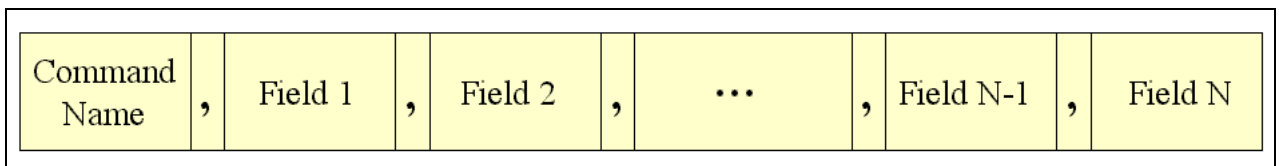
The DS2DC protocol works with text strings, directly sent and received through the TCP/IP protocol (via the activation of a TCP socket linking the DS2DC Client and the DS). Those text strings will be called "commands" hereinafter. Each command is composed of text fields, which are separated by commas. Each field contains a specific portion of the information included in the command. The first field is an uppercase text label, identifying the purpose of the command. That label will be referred as the "command name" hereinafter. Figure 21 illustrates the general structure of a DS2DC command.

Depending on their directionality, the commands can be classified in two groups:

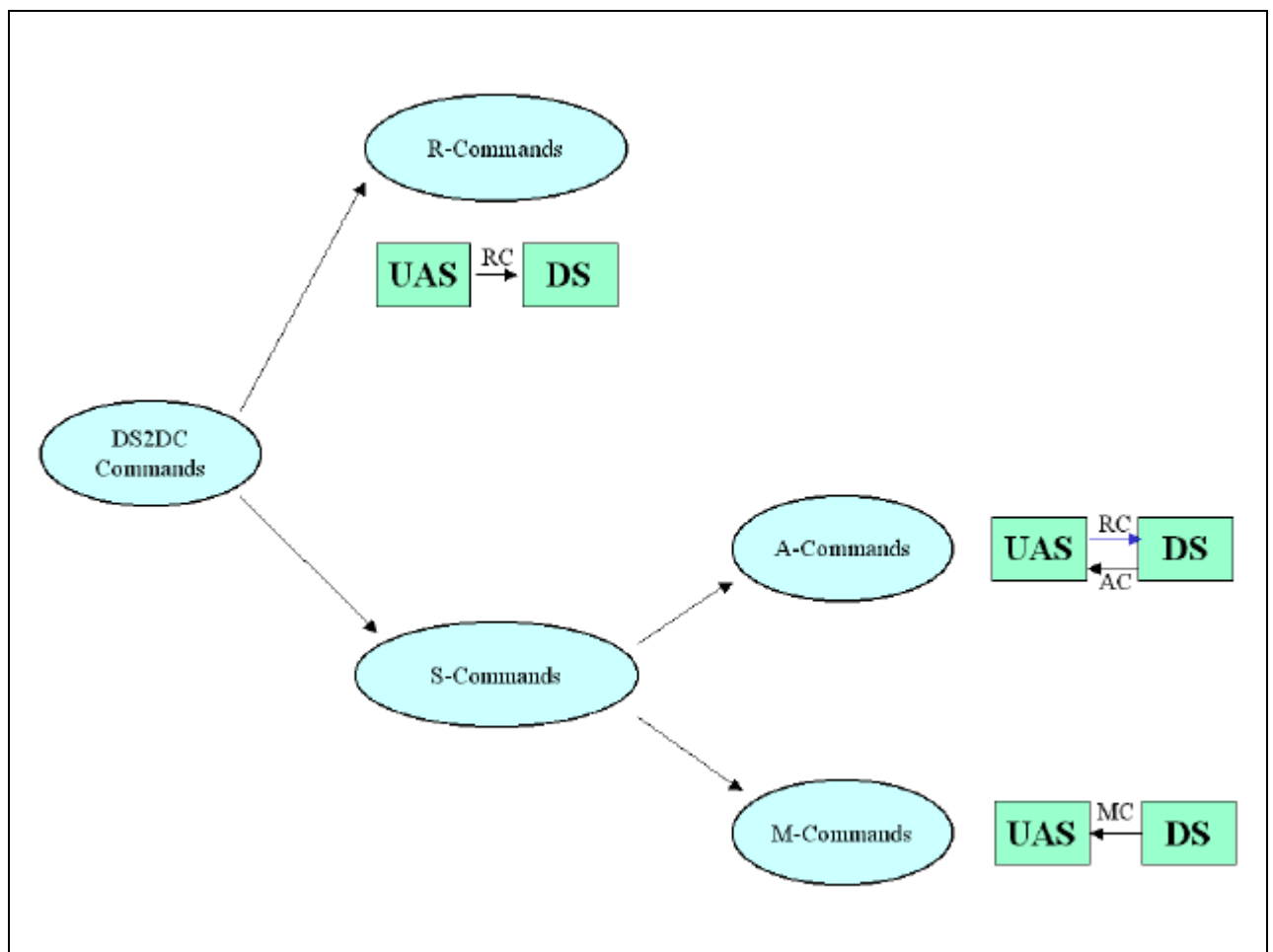
- **The commands going from the UAS to the DS.** Following the SISNET terminology, those commands receive the name of R-commands or RCs, since they usually **R**quest services to the DS.
- **The commands going from the DS to the UAS.** Following the SISNET terminology, those commands receiver the name of S-Commands or SCs, since they are used by the DS to provide a **S**ervice. The SCs can be broken down into two groups:

- **SCs sent as a response to previously sent RCs.** These commands receive the name of A-Commands or ACs, since they provide an Answer.
- **SCs spontaneously sent by the DS.** These commands receive the name of M-commands or MCs, since they can usually be understood as Messages coming from the server.

That classification of the DS2DC commands is illustrated in Figure 22.



*Figure 21. General structure of a DS2DC command*



*Figure 22. Types of DS2DC commands*

There is a relationship between the command names and the command types. The naming rules are the following:

- **RCs:** The command names always start with an alphabetic character.
- **MCs:** the command names always start by a '\*' character.
- **ACs:** the command names start by a '\*' character, followed by the command name which is being answered. The only exception to this rule is the '\*ERR' command. For instance, the AC associated with a MC called 'MSG' will be '\*MSG' (or '\*ERR' in case of error).

Looking at those rules, it is easy to conclude that the '\*' character indicates DS to UAS direction, and its absence indicates UAS to DS direction. The knowledge presented in this Section totally covers the DS2DC protocol theory. The next Section goes to the practical field, presenting all the available DS2DC commands.

#### 4.2.2 Available DS2DC Commands

The following list explains the purpose of the currently available DS2DC commands, using the classification introduced in Section 4.2.1:

- **R-Commands:**
  - **MSG.** Asks the DS for an EGNOS message. This command should be sent each second. Trying to send it with a higher transfer rate may derive into communication errors, due to the limitations of each particular link to SISNET. If the DS is not able to answer this command, an '\*ERR' command will be received.
  - **AUTH.** This command supports the new mandatory authentication process, which is previous to any interaction with the DS. For further information on the authorisation process, see Section 4.3.
- **A-Commands:**
  - **\*MSG.** This command answers the request of an EGNOS message. Its fields include the most up-to-date EGNOS message, characterized by the GPS time and week. The EGNOS message is provided in hexadecimal format, using the structure shown in Figure 23. The message itself contains 63<sup>1</sup> hexadecimal digits (252 bits). The EGNOS messages contain 250 bits, so that the last 2 bits must be ignored. The checksum can be used by the UAS for detecting transmission errors. Note the

---

<sup>1</sup> Due to several aspects of the receiver configuration at the BS, the received message could have 64 hexadecimal digits. In that case, the last digit is always set to zero and must be ignored.

EGNOS messages contain a 24-bit CRC code [12] that allows detecting and correcting errors.

- **\*AUTH.** A positive answer to an authentication command (AUTH). Indicates the DS2DC Client has permission to interact with the DS.
- **\*ERR.** This command informs about an error. Its contents include an error code and a text string explaining the error. The error code can be used for easily identifying the origin. In the first prototype versions of SISNET, a '\*ERR' MC indicates any kind of error.
- M-Commands:
  - **\*TXT.** Text messages received from the DS. This command allows the SISNET operators to broadcast informative text strings to the users. The normal response to this command is to present the contained string, using the available output device (PC monitor, LCD display, etc.).

ESA is working on adding new capabilities to the DS2DC protocol. As a probe of that, the following list shows some commands reserved for future use of those new features:

- R-Commands:
  - **POS.** This command will send the EGNOS-corrected position of the user to the DS, including an identifier number, which is unique for each user. The DS will store this information, allowing the development of a big amount of applications. For instance, it will be possible to draw the trajectory of the user on a map, and make it accessible through the WWW for tracking purposes.
  - **GETMSG.** This RC will allow obtaining a past EGNOS message (time of applicability less than current system time). The message to retrieve will be identified by MT number. This RC has interest for the MTs that are infrequently broadcast. For instance, MT18 is broadcast each 5 minutes. If the UAS needs that message and cannot allow waiting 5 minutes, the 'GETMSG' command will return the last MT18 broadcast by EGNOS.
  - **TROPO.** This command will request meteorological parameters involved in the calculation of troposphere corrections. The EGNOS-corrected position of the user will be included in the command. As indicated in MOPS [12], an EGNOS receiver interpolates on a table, in order to obtain an estimation of those parameters for its particular location. SISNET will be able to provide accurate measurements of those parameters, making use of Internet meteorological information servers.

- **GETTIME.** This command requests the DS to sent the current GMT time. The purpose is to set the user clock's time. This time reference cannot be used for precise operations, since the network delay adds some offset.
- **GETDATE.** Request the DS to send the current date.
- A-Commands:
  - **\*TROPO.** This command will return the meteorological parameters needed to improve the troposphere corrections.
  - **\*GETMSG.** The answer to the 'GETMSG' command. Contains the requested past message.
  - **\*GETTIME.** The DS uses this command to send the current GMT time.
  - **\*GETDATE.** The DS uses this command to send the current date.

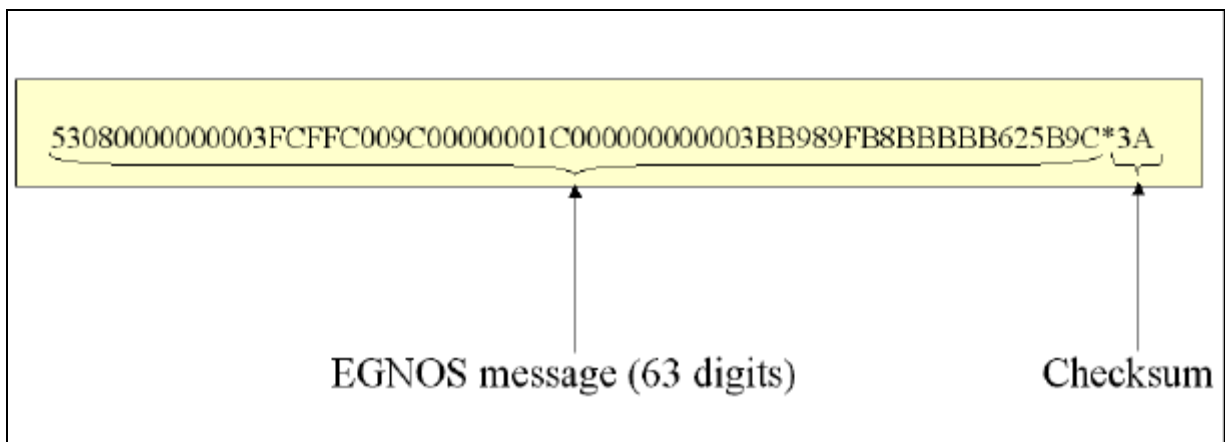


Figure 23. General structure of the EGNOS messages provided by SISNET.

The number of DS2DC commands will be continuously growing and, hence, being reflected into new revisions of this Document. Those commands are exclusively managed by ESA. Any suggestion of new commands is welcome, and will be studied by ESA. Sending commands not included in this Document to the DS is strictly prohibited.

A detailed explanation of the format of each available DS2DC command is presented in Appendix A. The commands reserved for future use are not included, since they are under study and development.

### 4.3 The Authentication Process

After opening a connection to the Data Server, the Client Socket must maintain an authentication dialog with the DS, in order to obtain permission for accessing the SISNeT services. As introduced in Section 4.2 (the previous Section), the authentication protocol relays on the use of the "AUTH" R-Command, and its associated "\*AUTH" A-Command. The complete sequence of actions is the following:

1. The DS2DC Client establishes a connection with the DS.
2. The DS2DC Client sends an AUTH command, containing the private information corresponding to a valid account, i.e. user name and password.
3. If the authentication data correspond to a valid account, the Data Server returns an \*AUTH command, indicating that the access to the DS is allowed. The DS2DC client can then start its normal interaction with the DS.
4. If the authentication data is not correct, the DS returns an \*ERR message containing error code number 2. This message indicates the access to the DS is denied. The DS disconnects the DS2DC client immediately after.

Note the authentication process is mandatory before any interaction with the DS, and must be respected exactly as shown above. The following remarks shall be taken into account:

- Obviously, no user can access the Data Server without having a valid SISNeT account. SISNeT accounts are managed by ESA, and must be requested through contacting the ESA SISNeT project team ([ftoran@esa.int](mailto:ftoran@esa.int)). A specific agreement for the use of SISNeT by the future client will then be set up.
- If the authentication phase is repeated certain number times without success, the DS will apply a penalty to the client, rejecting its connection requests during a certain number of hours.
- After connecting to the DS, the client socket must send an AUTH command. If the DS2DC Client tries to send other DS2DC commands before successfully passing the authentication phase, the DS will return an \*ERR command, containing error code set to 1. This message indicates the authorisation process is required. The DS disconnects the client immediately after. If this process is repeated several times, the DS applies a penalty, rejecting connections from the client for a certain number of hours.
- The user name is limited to 15 characters.
- The password is limited to 8 characters.

- The authentication protocol does not avoid the possible future release of public SISNeT Data Servers. In this particular case, any user will be able to access the DS through common authentication data, which will be made public. A hybrid version of the Data Server is also possible, allowing the public access through common authentication data, as well as private accounts (which will usually have some privileges in terms of performance).

#### 4.4 The SINCA compression algorithm

The SISNET Compression Algorithm (SINCA) is a simple technique, used by the DS to improve the communications speed when performing the most critical task: sending the EGNOS messages through the '\*MSG' command. If the EGNOS message were not compressed, its length into that command would always be 66 (or even 67) bytes, as shown in Figure 23. Using the SINCA algorithm, that length is frequently reduced down to 16 bytes, or even less. The compression rate depends on the contents of each concrete message.

In order to know how to decompress the messages, it is necessary to start by knowing how the messages are compressed. The SINCA compression algorithm is illustrated in Figure 24.

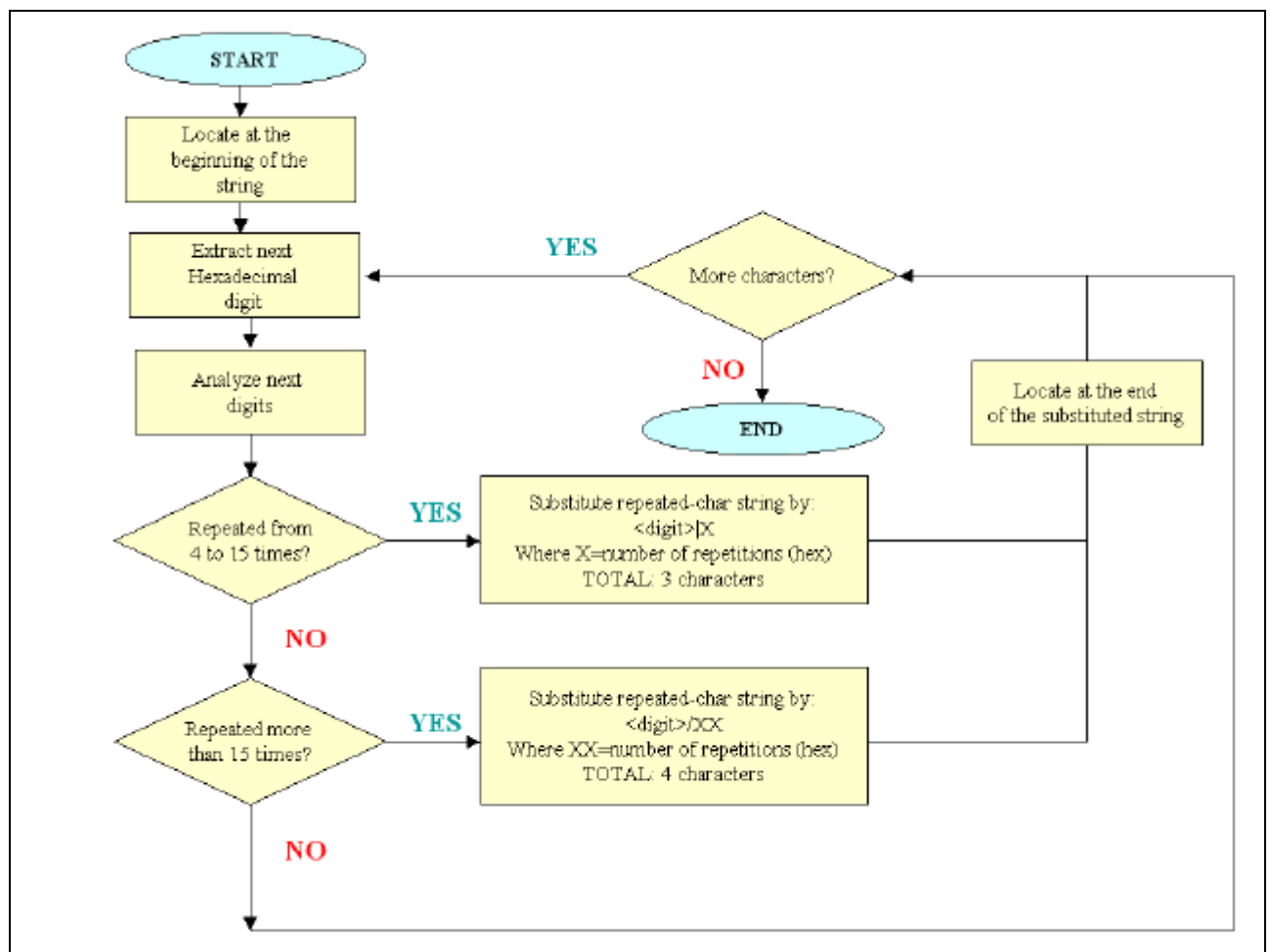


Figure 24. SINCA compression algorithm.



The process consists on evaluating each digit of the hexadecimal string. If the digit is repeated from 3 to 15 times, all the repetitions are substituted by:

$$\langle \text{repeated\_digit} \rangle | X$$

Where X is a hexadecimal digit indicating the number of repetitions. For instance, if the following sub-string is found:

AAAAAAAAAAAAA

That string will be substituted by:

A|C

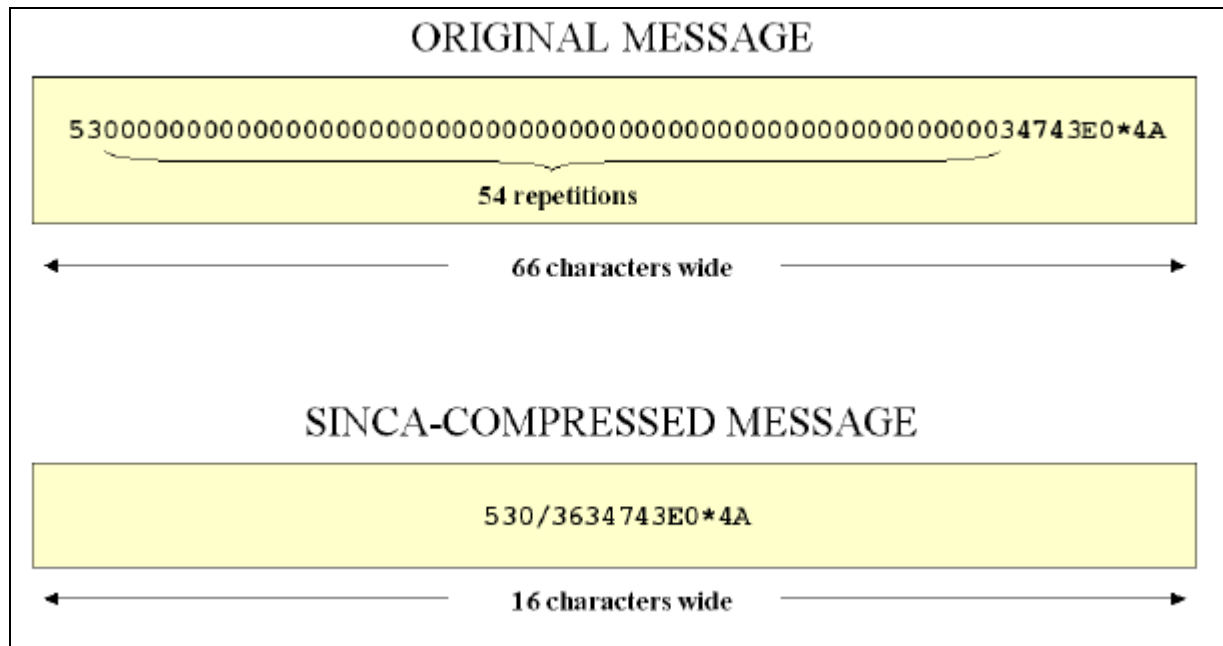
Since 'A' is repeated 12 times. Note just one digit is used to indicate the number of repetitions. That is the reason to allow a maximum of 15 repetitions. A minimum of 4 repetitions has been established, since the compressed string always has three characters. Otherwise, no compression would be achieved.

And, what happens if the digit is repeated more than 15 times? The way to compress is the same, but indicating the number of repetitions with two digits, and using the '/' separator:

$$\langle \text{repeated\_digit} \rangle / XX$$

The reason to use a different separator is to ease the decompression. The '|' character indicates to read only the next character. On the other hand, the '/' character indicates to read the next two characters.

Figure 25 shows an example of SINCA compression, performed over a real EGNOS MT0. The EGNOS message length goes from 66 characters down to only 16 characters. That means the message has been reduced to a 24.2% of the original size.



*Figure 25. Example of SINCA compression.*

**4.5 SINCA decompression algorithm**

Once the process of compression is clear, the SINCA decompression algorithm is quite easy to implement. This algorithm must be implemented by any UAS developer, in order to apply it to the compressed EGNOS messages included inside the '\*MSG' command. Figure 26 illustrates how to implement such an algorithm.

The process consists on evaluating each character of the compressed string. When an 'X|Y' string is found, that string is replaced by 'XX...X', where 'X' has been repeated Y times (note Y is an hexadecimal digit). When a 'X/YY' string is found, that string is replaced by 'XX...X', where 'X' has been repeated YY times (note YY is a two-digit hexadecimal number). The algorithm ends when reaching the end of the compressed string.

Figure 27 shows an example of SINCA-decompression, based on a real message broadcast by the ESTB. The original string was 26 characters wide. After decompression, the resulting string is 66 characters wide, i.e. 2.54 times longer.

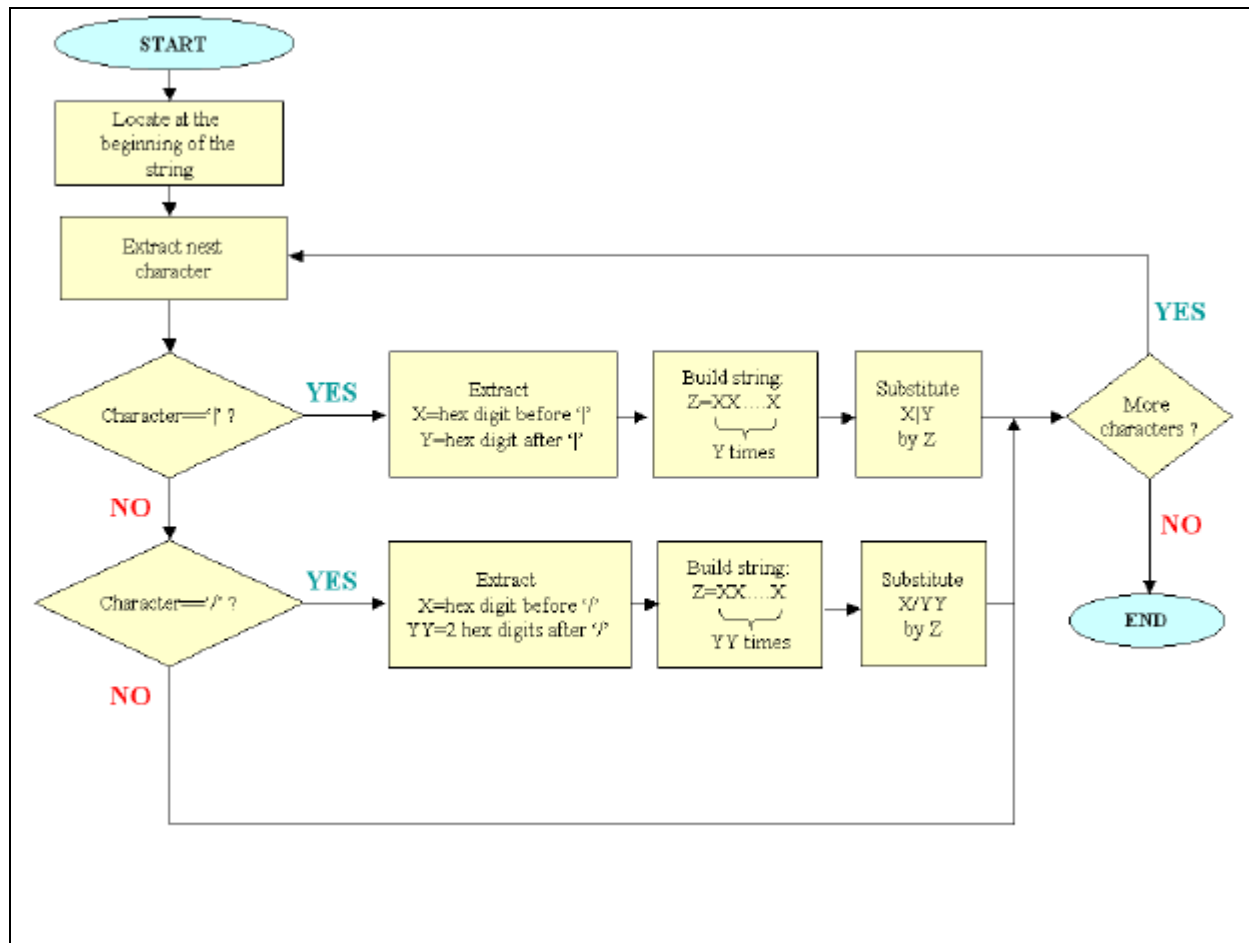


Figure 26. SINCA decompression algorithm.

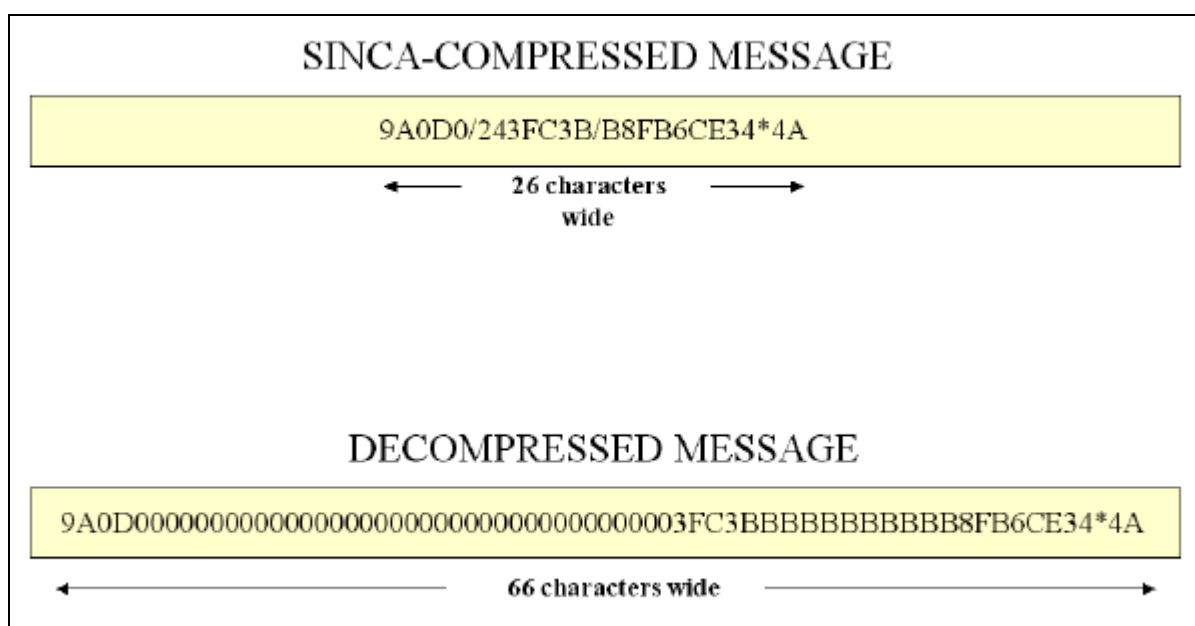


Figure 27. Example of SINCA decompression.

## 4.6 Summary

This Chapter has added more technical details to the concepts introduced in Chapter 3. The knowledge needed for the development of the DS2DC client (i.e. the most critical and restricted part of the UAS) has been introduced.

Concretely, the following information has been provided:

- **A technical description of the DS2DC protocol**, including the fundamental theory and the available DS2DC commands (see Appendix A). The DS2DC commands currently under development have been introduced as well.
- **A technical description of the SINCA compression algorithm**. Firstly, the compression algorithm has been described, since this knowledge is necessary to understand how to decompress the information contained in the '\*MSG' commands. Then, the decompression algorithm has been explained. The text has been illustrated through flow diagrams and examples.

Chapter 5 (the next Chapter) goes into a lower abstraction level by showing source code snippets, related to the development of the UAS.

## 5. PROGRAMMING TIPS AND SOURCE CODE EXAMPLES

### 5.1 Introduction

The purpose of this Chapter is to provide some technical inputs to the UAS developer, in order to ease the work of programming. The reading of the previous Sections is necessary, in order to understand the source code.

Each Section covers a specific facet of the UAS development, providing source code and its description. The source code is written in C++ language, adapted to the C++ Builder [14] development tool. However, the code can be easily ported to other languages such as Delphi or Visual Basic with no excessive modifications.

All the examples can be improved in several ways (e.g. enhanced error handling code). They are presented with only one objective: illustrating and teaching the basics of the UAS development, and also constituting a potential starting point for the developer.

Note the source code examples do not implement the authentication protocol, introduced in early February 2002 through the AUTH and \*AUTH commands. Nevertheless, the code can be easily upgraded, by simply inserting an event handler, which must respond to the socket connection event and take the appropriate actions. In the C++ Builder examples shown in this Chapter, the event to handle is *OnClientConnect*.

### 5.2 Implementation of the Client Socket

One of the vital parts of the DS2DC Client is the Client Socket (CS). Borland C++ Builder (BCB) - in its version 5.0 - presents a very appropriate component for the implementation of the CS, called *TClientSocket*. That component is located under the "Internet" tab of the components palette. Its objective is to open a TCP connection to a Server Socket (SS), and manage the communications at the level of the TCP protocol layer.

Figure 28 shows a *TClientSocket* component, placed on the main form of a BCB application. Several other components have been also placed on the form, in order to prepare a practical example. Two edit boxes allow setting the IP address of the DS and the port to use. Two buttons allow connecting to /disconnecting from the DS. The table in Figure 28 shows each component's class and name.

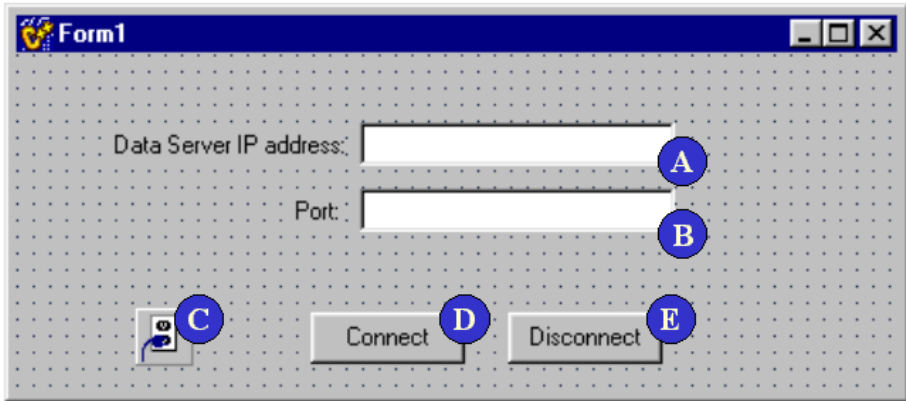
First of all, the CS must be able to connect to the DS, using the "Connect" button event handler:

```

void __fastcall TForm1::BtnConnectClick(TObject *Sender)
{
    //Set the Data Server IP and port
    Client->Address=IP->Text;
    Client->Port=Port->Text.ToInt();

    //Connect to the DS
    Client->Open();
}

```



ID	Class	Name
A	TEdit	IP
B	TEdit	Port
C	TClientSocket	Client
D	TButton	BtnConnect
E	TButton	BtnDisconnect

Figure 28. TClientSocket example.

The disconnection is implemented in a similar way:

```
void __fastcall TForm1::BtnDisconnectClick(TObject *Sender)
{
    //Disconnect from the DS
    Client->Close();
}
```

The communications can be managed through the events of the *TClientSocket* component:

- **OnClientError**. Indicates that an error has occurred during the communications.
- **OnClientDisconnect**. Triggered when the connection is effectively closed.
- **OnClientLookup**. Triggered just before trying to find the SS.
- **OnClientConnecting**. Thrown when the CS finds the SS.
- **OnClientConnect**. Indicates that the CS is connected to the SS.

The event handlers can be written as follows:

```
void __fastcall TForm1::ClientConnect(TObject *Sender,
    TCustomWinSocket *Socket)
{
    Caption="Connected to the Data Server";
}
//-----
void __fastcall TForm1::ClientConnecting(TObject *Sender,
    TCustomWinSocket *Socket)
{
    Caption="Connecting to the Data Server...";
}
//-----
void __fastcall TForm1::ClientDisconnect(TObject *Sender,
    TCustomWinSocket *Socket)
{
    Caption="Disconnected from the Data Server";
}
```

```
//-----  
void __fastcall TForm1::ClientError(TObject *Sender,  
    TCustomWinSocket *Socket, TErrorEvent ErrorEvent, int &ErrorCode)  
{  
    Caption="Communications error...";  
    Client->Close();  
    ErrorCode=0;  
}  
//-----  
void __fastcall TForm1::ClientLookup(TObject *Sender,  
    TCustomWinSocket *Socket)  
{  
    Caption="Looking for the Data Server...";  
}  
}
```

Those handlers just inform about the status of communications, using the form's caption. In the case of detecting a communications error, the CS is disconnected from the SS, using the *Close* method (otherwise, the program can become unstable). The reason to set the *ErrorCode* argument to zero is to avoid triggering an exception. The exception can be allowed by simply deleting the last line of the *ClientError* event handler.

### 5.3 Requesting and obtaining the EGNOS messages

Taking the example shown in Section 5.2 as a baseline, it is easy to implement code to handle the 'MSG' and '\*MSG' commands, i.e. to request and receive the EGNOS messages in real time.

The form needs small modifications:

- Adding a component able to show the received EGNOS messages;
- Adding a timing component, which must force requesting an EGNOS message each second.

Figure 29 shows a possible user interface responding to those modifications. A *TStaticText* component (identified as "Msg") is used for showing the EGNOS messages, while timing is controlled by a *TTimer* component (identified as "Timer", with its *Interval* property set to 900 milliseconds and the *Enabled* property set to false).



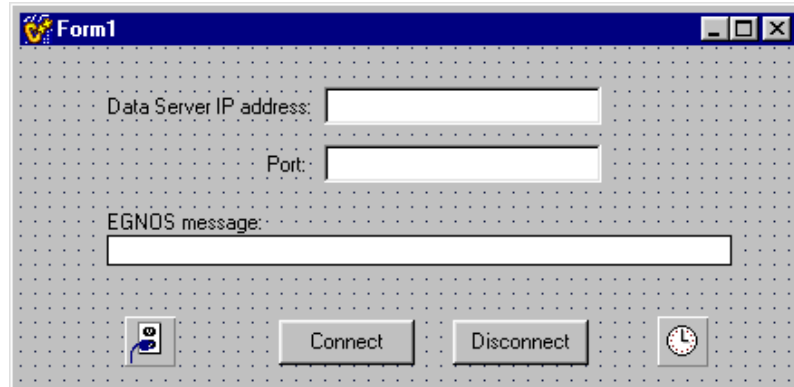


Figure 29. Changes to the main form for the reception of EGNOS messages.

The timer must start working when the connection to the DS is effective, and stop when disconnecting from the DS. This is achieved through some enhancements to the source code:

```
void __fastcall TForm1::ClientConnect(TObject *Sender,
    TCustomWinSocket *Socket)
{
    Caption="Connected to the Data Server";
    Timer->Enabled=true;
}
```

```
void __fastcall TForm1::BtnDisconnectClick(TObject *Sender)
{
    //Disconnect from the DS
    Timer->Enabled=false;
    Client->Close();
}
```

Each 900 milliseconds, the *TTimer* component triggers the *OnTimer* event. In that moment, an EGNOS message must be requested, using an 'MSG' command:

```
void __fastcall TForm1::TimerTimer(TObject *Sender)
{
    //Request an EGNOS message
    Client->Socket->SendText("MSG");
}
```

The DS will answer using a '\*MSG' command. When that command arrives to the UAS, the *OnClientRead* event of *TClientSocket* is thrown. The event handler must process the command, extract the EGNOS message and show it on the form:

```
void __fastcall TForm1::ClientRead(TObject *Sender,
    TCustomWinSocket *Socket)
{
    //Read the answer coming from the DS
    AnsiString Command=Client->Socket->ReceiveText();

    //Extract the EGNOS message (SINCA-compressed)
    AnsiString EGNOSMsg=GetMsg(Command);

    //Decompress the message
    EGNOSMsg=Decompress(EGNOSMsg);

    //Show the decompressed message on the main form
    Msg->Caption=EGNOSMsg;
}
```

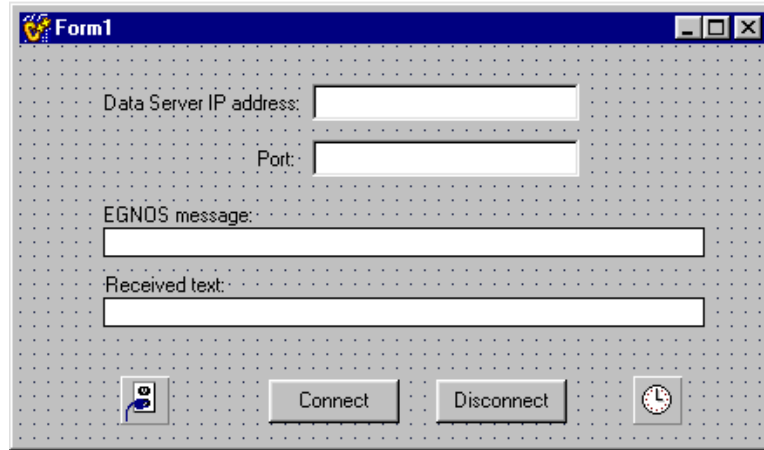
The *GetMsg* method must be able to analyse the received '\*MSG' command, and extract the EGNOS message (compressed version) from it. The *Decompress* method must implement the SINCA decompression algorithm, in order to restore the original EGNOS message.

The source code can be improved in several ways. For instance, it is possible to make the communications work as fast as possible, instead of using a timer.

#### **5.4 Displaying the text messages broadcast by SISNET**

The source code presented in Section 5.3 can be easily enabled to work with the '\*TXT' command, i.e. enabled to receive the text messages broadcast by SISNET.

The user interface only needs a field able to show the received text. Figure 30 shows a possible modification of the main form. A *TStaticText* component (identified as "TxtMsg") has been added, in order to show the received text.



The screenshot shows a Windows application window titled 'Form1'. It contains four text input fields: 'Data Server IP address:', 'Port:', 'EGNOS message:', and 'Received text:'. Below these fields are two buttons labeled 'Connect' and 'Disconnect', and a small clock icon on the right side of the form.

Figure 30. Enhancement of the main form for receiving text messages.

The source code requires small modifications. The code handling the '\*MSG' command can be put into a new method (*HandleMSG* method). The same can be done with the code handling the '\*TXT' command (*HandleTXT* method). The resulting code is the following:

```

AnsiString __fastcall TForm1::HandleMSG(AnsiString Command)
{
    //Extract the EGNOS message (SINCA-compressed)
    AnsiString EGNOSMsg=GetMsg(Command);

    //Decompress the message
    EGNOSMsg=Decompress(EGNOSMsg);
    //Show the message on the form
    Msg->Caption=EGNOSMsg;
}

void __fastcall TForm1::HandleTXT(AnsiString Command)
{
    //Extract the text from the *TXT command
    AnsiString Txt=GetTxt(Command);

    //Show the text on the main form
    TxtMsg->Caption=Txt;
}

```

The *GetTxt* method must analyse the '\*TXT' command, returning the content of the second field, i.e. the text message. The final step is to integrate the *HandleMSG* and *HandleTXT* methods into the *ClientRead* event handler:

```
void __fastcall TForm1::ClientRead(TObject *Sender,
    TCustomWinSocket *Socket)
{
    //Read the answer coming from the DS
    AnsiString Command=Client->Socket->ReceiveText();

    //Analyse the command type and execute the appropriate code
    if( IsMSG(Command) )
        HandleMSG(Command);

    else if( IsTXT(Command) )
        HandleTXT(Command);

    else
    {
        ShowMessage("The Data Server has sent an invalid command");
        return;
    }
}
```

The *IsMSG* method returns true only if the argument is a valid '\*MSG' command. The *IsTXT* method does the same with respect to the '\*TXT' command.

## 5.5 Summary

This chapter has gone into the lowest abstraction level of the UAS development. Some source code has been shown and described, covering important aspects of the UAS development. All the examples can be taken as a baseline for improving them in several ways. In fact, the purpose is just to illustrate a possible way to perform concrete programming tasks, but not to provide ready-to-integrate source code.

## 6. REFERENCES

1. L.Gauthier, P.Michel, J. Ventura-Traveset and J.Benedicto, "EGNOS: the first step of the European contribution to the Global Navigation Satellite System," *ESA Bulletin*, No. 105, Feb. 2001
2. H. Secretan, J. Ventura-Traveset, F. Toran, G. Solari and S. Basker, "EGNOS System Test Bed Evolution and Utilisation," *ION GPS 2001*, Salt Lake City (USA), September 2001.
3. F. Toran, J. Ventura-Traveset and J.C. de Mateo, "The ESA SISNET Project: Real-Time Access to the EGNOS Services Across the Internet," *Seventh International Workshop on Digital Signal Processing Techniques for Space Communications*, Sesimbra (Portugal), October 2001. Available at <http://www.esa.int/estb>
4. F. Toran, J. Ventura-Traveset and J.C. de Mateo, "Internet-based Satellite Navigation receivers using EGNOS: the ESA SISNET project," *ESA Workshop on Satellite Navigation User Equipment Technologies (NAVITEC)*, Noordwijk (The Netherlands), December 2001, in press. Available at <http://www.esa.int/estb>
5. "SISNET: Making EGNOS Available Over the Internet," *ESTB News*, ESA Newsletter, Issue 2, page 4, September 2001.
6. F. Toran, J. Ventura and J.C. de Mateo. "The ESA SISNET Project: Real-Time EGNOS Services through the Internet," *ESA Journal Preparing for the Future*, in press.
7. F. Toran, J. Ventura-Traveset and J.C. de Mateo, "ESPADA 3.0: An innovative EGNOS Simulation Tool Based on Real Data," *ESA Journal preparing for the Future*, in press.
8. A. García, C. Garriga, P. Michel and J. Ventura-Traveset, "EGNOS Simulation Tool for Performance Assessment and Design Analysis (ESPADA)," *ESA Journal preparing for the Future*, Vol. 7, No.4, December 1997.
9. F. Toran, "Advanced Simulation Tool for Satellite Navigation: from Radio Frequency to Positioning Using Real Data," Research Work Essay, Directed by Dr. Diego Ramirez (University of Valencia) and Dr. Javier Ventura-Traveset (ESA), July 2001.
10. ESA, *EGNOS System Test Bed Website*. <http://www.esa.int/navigation/estb>
11. F. Torán and J. Ventura-Traveset, "Justification of the ESA SISNeT Concept through Advanced Modelling of User Masking Effects," ESA internal note, February 2002.

12. RTCA, "*Minimum Operational Performance Standards for Global Positioning System / Wide Area Augmentation System Airborne Equipment*," Ref. RTCA/DO-229B, October 6, 1999
13. N. Suard, "*ESTB SIS User Interface Description*," Issue 0, Rev. 1, Ref.: E-TN-ITF-E31-0008-ESA.
14. Inprise, "*Borland Community*," <http://community.borland.com>
15. "*Standard Positioning Service GPS*," US DoD, Washington, October 2001.
16. F. Toran and J. Ventura-Traveset. "The ESA SISNET Project: Real-Time Access to the EGNOS Services across the Internet," *2<sup>nd</sup> ESA ESTB Workshop*, Nice (France), November 2001. Available at <http://www.esa.int/estb>.
17. H. Secretan, J. Ventura-Traveset, F. Toran, G. Solari and S. Basker, "EGNOS System Test Bed Evolution and Utilisation," *NAVITEC Conference*, Noordwijk (The Netherlands), December 2001. Available at <http://www.esa.int/estb>.
18. F. Toran, J. Ventura-Traveset and J.C. de Mateo, "Satellite Navigation and the Internet: Introducing SISNET Technology," to appear on Dr. Dobb's Journal, March 2002. Featured article of March 2002. Article available online at Dr. Dobb's Journal Official Website, at <http://www.ddj.com/documents/s=4069/ddj0203a/0203a.htm>

## 7. LIST OF ACRONYMS

AC	A-Command
ADC	Analysis of DS2DC Commands
AL	Alarm Limit
BS	Base Station
BSS	Base Station Software
DA	Decompression Algorithm
DS	Data Server
DSS	Data Server Software
DS2DC	Data Server to Data Client protocol
EGNOS	European Geostationary Navigation Overlay Service
ESTB	EGNOS System Test Bed
FTP	File Transfer Protocol
GEO	Geostationary Satellite
GUI	Graphical User Interface
JPEG	Joint Photographic Experts Group
MC	M-Command
MT	Message Type
PRC	Pseudo-Range Correction
PRN	Pseudo-Random code number
RC	R-Command
RCS	Receiver Control Software
SC	S-Command
SDS	SISNET Data Server
SE	Software Engineering
SINCA	SISNET Compression Algorithm
SIS	Signal In Space
SISNET	Signal In Space through the InterNET
SUI	SISNET User Interface
TCS	TCP/IP Client Socket
UAS	User Application Software
UDRE	User Differential Ranging Error
UID	User Interface Document
VPL	Vertical Protection Limit
WWW	World Wide Web

## APPENDIX A. DS2DC Command Reference.

### AUTH Command

---

#### Type

R-Command

#### Purpose

This R-Command is the first to be sent after connecting to the Data Server. It is the base of the SISNeT authentication protocol, providing information on a particular SISNeT account (i.e. username and password).

#### Format

<b>AUTH</b>	,	User	,	Password
-------------	---	------	---	----------

#### Fields:

Field	Comments
AUTH	The command name
User	Username corresponding to a valid SISNeT account.
Password	The corresponding password.



**\*AUTH Command**

---

Type

A-Command

Purpose

This command gives a positive answer to the AUTH command, indicating that a remote user is authorised to use the Data Server. If the user is not authorised, an \*ERR command will be sent instead.

Format

<b>*AUTH</b>
--------------

Fields:

Field	Comments
*AUTH	The command name

**\*ERR Command**

---

Type

A-Command. Note this command is an exception to the naming rule for A-Commands. It uses the same command name for answering any R-Command.

Purpose

This command informs about any error thrown during the communications between the UAS and the DS.

Format

<b>*ERR</b>	,	Error code	,	Error message
-------------	---	------------	---	---------------

Fields:

Field	Comments
*ERR	The command name.
Error code	A number identifying the error. It is used by the UAS to easily identify what has happened and how to proceed.
Error message	A text string explaining the error.

Remarks:

See Appendix B for a comprehensive reference of error codes and messages.

## MSG Command

---

### Type

R-Command

### Purpose

This command requests the DS to send the most up-to-date EGNOS message.

### Format

**MSG**

### Fields:

Field	Comments
MSG	The command name

### Remarks:

- The DS shall ignore any field included in this command.
- If (for any reason) the DS cannot provide the service to the UAS, a '\*ERR' message will be returned.
- The UAS is expected to send this command once per second. However, higher rates can be employed, depending on the concrete characteristics of the communications link.

**\*MSG Command**Type

A-Command

Purpose

The DS uses this command to answer "MSG" commands. The most up-to-date EGNOS message is sent, complemented with the corresponding GPS time and week.

Format

<b>*MSG</b>	,	GPS Week	,	GPS time	,	EGNOS Message (hex)
-------------	---	----------	---	----------	---	---------------------

Fields:

Field	Comments
*MSG	The command name
GPS Week	The GPS week corresponding to the EGNOS message
GPS time	The GPS time corresponding to the EGNOS message
EGNOS message (hex)	The EGNOS message, compressed through the SINCA algorithm (see Chapter 4)

Remarks:

- The GPS week number is represented using 10 bits, so that the number ranges from 0 to 1023. At the time of this writing (October 2001), the first rollover was crossed. Since SISNET can use different kinds of EGNOS receivers, and each receiver works with a different GPS week format, it is possible to receive a GPS week that does not take rollover into account. For instance, if the UAS receives a week number of 78, that means the real week is  $78+1024=1102$ . In the other side, if the UAS receives week 1102, no processing is needed. The developer of the UAS must take this effect into account.

**\*TXT Command**

---

Type

M-Command

Purpose

The DS uses this command to broadcast text messages to all the users connected to SISNET. The UAS should be able to present the received text messages, normally using a GUI. Some possible usages of those messages are the following:

- Informing the users about SISNET-related events (e.g. new enhancements);
- Warning the users about a scheduled outage of the SISNET service;

Format

<b>*TXT</b>	,	Text string
-------------	---	-------------

Fields:

Field	Comments
*TXT	The command name
Text string	The text broadcast from the DS

**APPENDIX B. SISNeT Error Codes and Error Messages.**

<b>Error Code</b>	<b>Error Message</b>	<b>Comments</b>
1	Authorisation required	This error code is returned when the client application sends a DS2DC command without successfully passing the initial authentication process. After receiving this command, the client is disconnected from the DS. If the client maintains the same attitude (i.e. avoiding the authentication process), a penalty will be applied (e.g. access denied for a certain number of hours).
2	Access Denied	Indicates the user has not successfully passed the authentication process. Hence, the access to SISNeT is denied. If the user fails to pass the authorisation process more than N times, a penalty will be applied (N to be determined and susceptible of changes during the operational life of SISNeT)
3	Unknown DS2DC Command	Returned when the Data Server receives an unknown command. A penalty is applied if a given user continually sends unknown commands.
4	Command was not successfully completed	Indicates the Data Server was not able to complete the requested action. This can be due to several reasons, e.g. eventual failures. The user is encouraged to report this situation to the SISNeT team.

**IMPORTANT NOTE:** error codes 3 and 4 are not implemented at the time of this writing (February 2002).