RTCM STANDARD 10410.0

(RTCM Paper 200-2004/SC104-STD, Version 1.0)

FOR

NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL (Ntrip)

DEVELOPED BY
RTCM SPECIAL COMMITTEE NO. 104

SEPTEMBER 30, 2004

Radio Technical Commission For Maritime Services
1800 N. Kent St., Suite 1060
Arlington, Virginia 22209-2109, U.S.A.
E-Mail: info@rtcm.org
Web Site: http://www.rtcm.org

The Radio Technical Commission For Maritime Services (RTCM) is an incorporated non-profit organization, with participation in its work by international representation from both government and non-government organizations. The RTCM does not work to induce sales, it does not test or endorse products, and it does not monitor or enforce the use of its standards.

The RTCM does not engage in the design, sale, manufacture or distribution of equipment or in any way control the use of this standard by any manufacturer, service provider, or user. Use of, and adherence to, this standard is entirely within the control and discretion of each manufacturer, service provider, and user.

For information on RTCM Documents or on
participation in development of future RTCM documents contact:

Radio Technical Commission For Maritime Services
1800 N. Kent St., Suite 1060
Arlington, Virginia 22209-2109 USA

Telephone: +1-703-527-2000
Telefax: +1-703-351-9932
E-Mail: info@rtcm.org

RTCM STANDARD 10410.0
(RTCM Paper 200-2004/SC104-STD, Version 1.0)

FOR

NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL (Ntrip)

DEVELOPED BY
RTCM SPECIAL COMMITTEE NO. 104

SEPTEMBER 30, 2004

Radio Technical Commission For Maritime Services
1800 N. Kent St., Suite 1060
Arlington, Virginia 22209-2109, U.S.A.
E-Mail: info@rtcm.org
Web Site: http://www.rtcm.org

This page intentionally left blank.

ADDENDUM 1
to
RTCM RECOMMENDED STANDARDS
FOR
NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL
(Ntrip)
VERSION 1.0

RTCM Paper 200-2004/SC104-STD, (RTCM Recommended Standards for Networked Transport of RTCM Via Internet Protocol (Ntrip) Version 1.0, dated September 30, 2004) is revised as follows:

Assign RTCM Standard number 10410.0 to Version 1.0 of the RTCM Recommended Standards for Networked Transport of RTCM via Internet Protocol (RTCM Paper 200-2004/SC104-STD).

Replace pages 4-1, 4-2, 5-1, and A-1 with the enclosed pages 4-1, 4-2, 5-1, and A-1

(Pages above have been replaced in this document.)

TABLE OF CONTENTS

This page intentionally left blank.

PREFACE

Networked Transport of RTCM via Internet Protocol (Ntrip) is an application-level protocol that supports streaming Global Navigation Satellite System (GNSS) data over the Internet. Ntrip is a generic, stateless protocol based on the Hypertext Transfer Protocol HTTP/1.1. The HTTP objects are extended to GNSS data streams.

Ntrip is designed to disseminate differential correction data or other kinds of GNSS streaming data to stationary or mobile users over the Internet, allowing simultaneous PC, Laptop, PDA, or receiver connections to a broadcasting host. Ntrip supports wireless Internet access through Mobile IP Networks like GSM, GPRS, EDGE, or UMTS.

Ntrip consists of three system software components: NtripClients, NtripServers and NtripCasters. The NtripCaster is the actual HTTP server program, while NtripClient and NtripServer act as HTTP clients.

Ntrip is meant to be an open non-proprietary protocol. Major characteristics of Ntrip's dissemination technique are the following:

- It is based on the popular HTTP streaming standard; it is comparatively easy to implement when limited client and server platform resources are available.
- Its application is not limited to one particular plain or coded stream content; it has the ability to distribute any kind of GNSS data.
- It has the potential to support mass usage; it can disseminate hundreds of streams simultaneously for up to a thousand users when applying modified Internet Radio broadcasting software.
- Regarding security needs, stream providers and users are not necessarily in direct contact, and streams are usually not blocked by firewalls or proxy servers protecting Local Area Networks.
- It enables streaming over any mobile IP network because it uses TCP/IP.

This documentation describes Ntrip Version 1.0. Further Ntrip versions may be issued as the need arises.

This page intentionally left blank.

1 INTRODUCTION

Since the deliberate degradation of GPS signals, called Selective Availability, was turned off, stand-alone GPS receivers typically experience position errors of 10-15 meters, and these errors follow a random walk pattern. While this accuracy is adequate for a wide variety of applications, there are other applications where meter-level, or even centimeter-level, accuracy is desired. For such high-accuracy applications, a Differential Global Navigation Satellite System (DGNSS) can be employed successfully. A DGNSS improves the accuracy of position, velocity, and time by providing measurements or correction data from one or more reference stations to mobile receivers. The position of each reference station is accurately known, so its measurements act as a calibration for other nearby receivers, because the major error sources are common: satellite ephemeris and clock, tropospheric and ionospheric delay. The DGNSS accuracy degrades as the distance between user and reference receivers increases, so for some applications, information from multiple reference stations is desirable.

A DGNSS can utilize differential corrections for any GNSS (GPS, GLONASS, Galileo) to achieve meter-level accuracy, or it can use Real Time Kinematic (RTK) information to achieve decimeter or centimeter accuracy. The DGNSS data needs to be refreshed every few seconds to keep up with atmospheric changes.

RTCM-104 DGNSS standards are used worldwide. Many popular GNSS receivers accept RTCM-104 differential correction messages, and many special-purpose RTK receivers use the RTK messages. The RTCM SC-104 standards define messages that contain reference station and system data. These messages are typically broadcast over a transmission link and received by mobile users, which then apply the information contained in the messages to achieve high-accuracy operation. The data can be transmitted over any number of communication channels, e.g., via radio transmission (LF, MF, HF, UHF), or via a mobile communication network, using different communication protocols.

The introduction and deployment of wireless Internet capability has opened up the potential of disseminating these messages over the Internet. This document defines a standard for an HTTP-based protocol for the dissemination of DGNSS data (or other kinds of GNSS streaming data) to stationary or mobile receivers over the Internet. It enables simultaneous PC, Laptop, PDA, or receiver connections to a broadcasting host, via IP Networks such as GSM, GPRS, EDGE, or UMTS. The protocol development is the result of a feasibility study on real-time streaming of differential GPS corrections as described in [1]. Because the primary application will be the dissemination of RTCM SC-104 messages, the system as a whole presents a transport protocol that will be referred to herein as the Ntrip Protocol (Networked Transport of RTCM via Internet Protocol).

The basic data streaming is accomplished by TCP/IP protocol stack. Several demonstrations based on a plain Serial-to-TCP conversion of streaming data on the reference-side (server) and TCP-to-Serial re-conversion on the rover-side (client) have shown the suitability of the TCP/IP protocol for streaming data to mobile IP clients [2].

This page intentionally left blank.

2 SYSTEM CONCEPT

The Hypertext Transfer Protocol (HTTP) is designed as an application-level protocol for distributed collaborative hypermedia information systems, but it can also be used for linear streaming media. HTTP is primarily used for bulk traffic, where each object has a clearly defined beginning and end. Although widely used for IP streaming applications, which include the RTCM application, it is not designed for such uses. The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax, is defined in the protocol and transmitted via a TCP/IP connection. Client and server must understand HTTP request messages, and must answer with adequate HTTP respond messages.

Ntrip, which uses HTTP, is implemented in three programs: NtripClient, NtripServer and NtripCaster, where the NtripCaster is the real HTTP (splitter-) server. NtripClient and NtripServer act as HTTP client programs.

In its message format and status code, the NtripClient-NtripCaster communication is based on HTTP 1.1 communication [3], where Ntrip uses only non-persistent connections. The NtripServer-NtripCaster communication deliberately deviates from HTTP by a new message format, which is called “SOURCE”, and a new status-code, which is called “ERROR - Bad Password”.

A loss of the TCP connection between communicating system-components (NtripClient-NtripCaster, NtripServer-NtripCaster) will be automatically recognized by the TCP-sockets. This effect can be used to trigger software events such as an automated reconnection.

This Ntrip system (see Figure 1) consists of the following elements:

- NtripSources, which generate data streams at a specific location,
- NtripServers, which transfer the data streams from a source to the NtripCaster,
- NtripCaster, the major system component, and
- NtripClients, which finally access data streams of desired NtripSources on the NtripCaster.

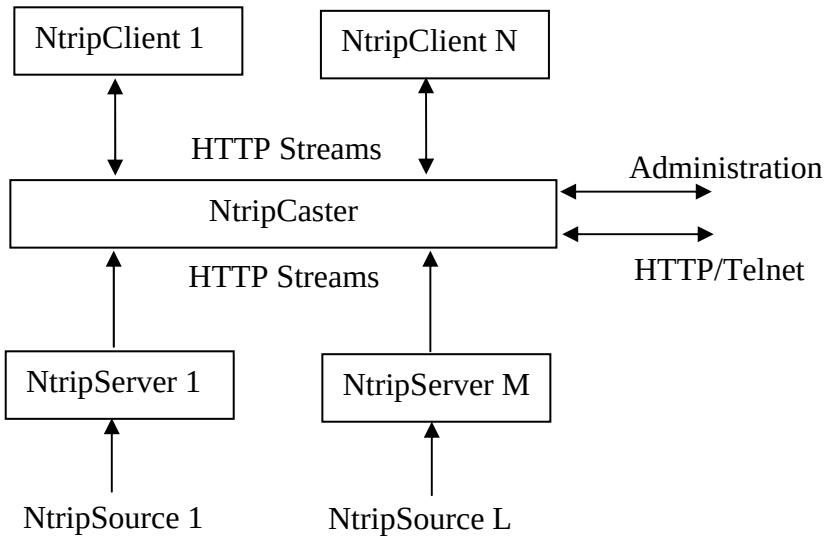

Figure 1. Ntrip Streaming System

NtripServers define a source ID called "mountpoint" for every streamed NtripSource. Several NtripClients can access the data of desired NtripSources at the same time by requesting a source by its mountpoint on the NtripCaster. If implemented in the NtripCaster program, authorized personnel may remotely control the NtripCaster via a password-protected Telnet session, or receive status information via a password-protected HTTP session using an Internet Browser.

An administrator running an NtripCaster is responsible for allowing new NtripServers to connect with new NtripSources. The administrator organizes all available NtripSources and defines all source IDs (mountpoints).

NtripClients must be able to choose an NtripSource by its mountpoint on the NtripCaster. Therefore a source-table is introduced into, and maintained on, the NtripCaster. Each record of this source-table contains parameters describing attributes of a data stream, a network of data streams, or an NtripCaster. Stream attributes (identifier, coordinates, format, nav-system, mountpoint, etc.) are defined at the NtripServer side for each NtripSource.

If an NtripClient sends an invalid or no mountpoint (no or not up-to-date source-table available for the client), the NtripCaster will upload an up-to-date source-table as an HTTP object instead of a GNSS data stream. Afterwards the NtripClient has the up-to-date source-table available and can connect to a GNSS data stream on the NtripCaster.

The Ntrip system depends on direct communication between the responsible administrators of NtripCasters and NtripServers (e.g. via email). They must specify the parameters characterizing an NtripSource/mountpoint in the source-table.

The RTCM SC-104 Committee is aware that the use of HTTP for Ntrip as described in this document differs from the HTTP standard. The protocol described here utilizes a pragmatic approach that incorporates IP streaming, similar to techniques employed by internet radio and video conferencing. The intended constituency for the RTCM information is the wireless mobile user community, which doesn't use proxy servers. While the protocol described here works with many proxy servers, their use should be avoided whenever possible.

3 SYSTEM ELEMENTS

3.1 GENERAL

A DGNSS reference station in its simplest configuration consists of a GNSS receiver, located at a well-surveyed position. Because this stationary-operated GNSS receiver knows where the satellites are located in space at any point in time, as well as its own exact position, the receiver can compute theoretical distance and signal travel times between itself and each satellite. When these theoretical values are compared to real observations, differences represent errors in the received signals. RTCM corrections are derived from these differences. Making these corrections available in real-time for mobile users is the major purpose of the Ntrip system elements, although Ntrip may be used as well for transporting other types of GNSS streaming data (such as RTK) over its system elements NtripServer, NtripCaster, and NtripClient.

3.2 NTRIPSOURCE

The NtripSources provide continuous GNSS data (e.g. RTCM-104 corrections) as streaming data. A single source represents GNSS data referring to a specific location. Source description parameters as compiled in the source-table specify the format in use (e.g. RTCM 2.0, RTCM 2.1, Raw), the recognized navigation system (e.g. GPS, GPS+GLONASS), location coordinates and other information.

Note: Every single NtripSource needs a unique mountpoint on an NtripCaster.

3.3 NTRIPSERVER

The NtripServer is used to transfer GNSS data of an NtripSource to the NtripCaster. Before transmitting GNSS data to the NtripCaster using the TCP/IP connection, the NtripServer sends an assignment of the mountpoint.

Server passwords and mountpoints must be defined by the administrator of the NtripCaster and handed over to the administrators of the participating NtripServers. An NtripServer in its simplest setup is a computer program running on a PC that sends correction data of an NtripSource (e.g. as received via the serial communication port from a GNSS receiver) to the NtripCaster.

The Ntrip protocol may be used for the transport of RTCM data of a virtual reference station following the so-called VRS concept. Based on data from a number of reference stations, RTCM corrections are derived for a virtual point at the users approximate position. Data for this virtual reference station represent a single NtripSource that can be transmitted by an NtripServer.

3.4 NTRIPCASTER

The NtripCaster is basically an HTTP server supporting a subset of HTTP request/response messages and adjusted to low-bandwidth streaming data (from 50 up to 500 Bytes/sec per stream). The NtripCaster accepts request-messages on a single port from either the NtripServer or the NtripClient. Depending on these messages, the NtripCaster decides whether there is streaming data to receive or to send.

An NtripServer could be a part of the NtripCaster program. If so, only the capability of receiving NtripClient messages has to be implemented into the combined NtripCaster/NtripServer. Built-in HTTP-based remote administration capability is an optional function.

3.5 NTRIPCLIENT

An NtripClient will be accepted by and receive data from an NtripCaster, if the NtripClient sends the correct request message (TCP connection to the specified NtripCaster IP and listening Port). With respect to the message format and status code, the NtripClient-NtripCaster communication is fully compatible to HTTP 1.1, but Ntrip uses only non-persistent connections.

4 SERVER COMMUNICATION

The NtripServer-NtripCaster communication extends HTTP by the additional message format "SOURCE" and the additional status-codes "ERROR - Bad Password" and "ERROR – Mount Point Taken or Invalid" or "ERROR – Already Connected". The password is not protected and therefore is based, as in the HTTP Basic Access Authentications scheme, on the assumption that the connection between the client and the server can be regarded as a trusted carrier.

The NtripServer must connect to the NtripCaster using the IP and specified listening port of the NtripCaster. This means that the NtripCaster must be "up and running" before any source can connect. Before transmitting GNSS data to the NtripCaster using the TCP/IP connection, the NtripServer must send an Ntrip server message in order to get access and to specify a mountpoint. The Ntrip server message is shown in Figure 2.

```
SOURCE <password> /<mountpoint> <CR><LF>
Source-Agent: NTRIP<product|comment><CR><LF>
STR: <STR-string><CR><LF>
<CR><LF>
<data>
```

Figure 2. Server Message Structure

<password> = Encoder password of the Caster
<mountpoint> = Caster mountpoint for the Source
<STR-string> = Data fields 3 to n of source-table record type "STR", optional
<product|comment> = Information about the source agent

String <STR-string> is meant to send source-table parameters to the NtripCaster. Including it in the server message is optional. When sending <STR-string>, it must contain all parameters describing the corresponding source-table record of type "STR" beginning with field number 3 up to field number n (see Source-table section). Data fields in <STR-string> are separated by field delimiter ";".

Example: An NtripServer sends the Ntrip server message:

```
⇒ SOURCE letmein /Mountpoint
  Source-Agent: NTRIP NtripServerCMD/1.0
```

If the password is correct, the NtripCaster will answer:

⇐ ICY 200 OK

The caster accepts data after sending the message “ICY 200 OK”. The NtripCaster will only accept data from sources with the valid encoder password. An administrator must pre-define this password on the caster. The password is coded as a plain ASCII string.

Sending a false password leads to the caster response message:

⇐ ERROR - Bad Password

The caster then automatically quits the TCP connection. If the declared mountpoint is already taken from another server or if the mountpoint is invald, the NtripCaster will answer:

⇐ ERROR – Mount Point Taken

If the declared mountpoint is invalid, the NtripCaster will answer:

⇐ ERROR – Mount Point Invalid

Due to compatibility reasons with older implementations two other possible answers have to be expected. Old versions of NtripCasters might still answer with either

⇐ ERROR – Mount Point Taken or Invalid

or

⇐ ERROR – Already Connected

For new implementations the unambiguous error reporting
“⇐ ERROR – Mount Point Taken “ and “⇐ ERROR – Mount Point Invalid”
should be implemented.

5 CLIENT COMMUNICATION

5.1 GENERAL

A client's request is designed as an HTTP message similar to the server message shown in Figure 2. Each client only needs to know the mountpoint of the desired data stream. The message for requesting a data stream is defined in Figure 3. The User-Agent request-header field must begin with the string NTRIP.

```
GET/<mountpoint> HTTP/1.0 <CR><LF>
User-Agent: NTRIP<product|comment><CR><LF>
Accept: */* <CR><LF>
Connection: close <CR><LF>
```

Figure 3. Client Message Structure

<mountpoint> = Caster mountpoint of requested source
<product|comment> = Information about the user agent originating the request

Example: A client sends (in the non protected case) the request message:

```
⇒ GET /BUCU0 HTTP/1.1
  User-Agent: NTRIP GNSSInternetRadio/1.2.0
```

For a valid request (desired NtripSource/mountpoint exists), the NtripCaster will answer:

```
⇐ ICY 200 OK
```

followed by the GNSS data

```
⇐ <GNSS data>
```

For an invalid request (desired NtripSource/mountpoint does not exist), the NtripCaster will answer:

```
⇐ SOURCETABLE 200 OK
```

followed by the source-table object

```
⇐ <Source-Table>
```

Via the information from the source-table, as maintained by the NtripCaster, NtripClients are enabled to choose the desired NtripSource/mountpoint from all available NtripSources/ mountpoints. An NtripClient can either store a source-table in memory (e.g. hard disk), or request a new source-table before requesting an NtripSource. The desired NtripSource can be chosen manually (e.g. based on identifier, nav-system, and format information) or by software (e.g. based on position, format, and nav-system information).

Requesting unavailable mountpoints (out-of-date source-table) will automatically result in the caster replying with an up-to-date source-table. Therefore, mountpoints, as a synonym for a specific NtripSource, must be unique on an NtripCaster. Using a four-character-ID followed by an integer value (e.g. BUCU0 for Bucharest) as mountpoint parameter is recommended.

An example for handling client messages in C programming language is given in Appendix C. This simple Linux/UNIX NtripClient program reads data from an NtripCaster and writes on standard output for further redirection of data to a file or COM-port.

5.1 BASIC AUTHENTICATION SCHEME

For billing purposes, the NtripSources/mountpoints can be password-protected on an NtripCaster. In this protected case the HTTP communication differs from the non-protected case.

Example:

The client will send (like in the non-protected case) the request message:

```
⇒ GET /BUCU1 HTTP/1.0
  User-Agent: NTRIP GNSSInternetRadio/1.2.0
```

The HTTP-server will answer a client request according to the HTTP Protocol [4] with

```
⇐ HTTP/1.0 401 Unauthorized
```

for invalid passwords and send a second message to the client:

```
⇐ Server: NtripCaster/1.0
  WWW-Authenticate: Basic realm="/BUCU1"
  Content-Type: text/html
  Connection: close
  <html><head><title>401 Unauthorized</title></head><body bgcolor=black text=white
        link=blue alink=red>
  <h1><center>The server does not recognize your privileges to the requested entity
        y/stream</center></h1>
  </body></html>
```

The client sends a second request message for the same mountpoint including the base64-coded user:password string to the caster:

```
⇒ GET /BUCU1 HTTP/1.0
  User-Agent: NTRIP GNSSInternetRadio/1.2.0
  Authorization: Basic aHVnb2JlbjpodWdvYmVuMTIz
```

The NtripCaster will answer

```
⇐ ICY 200 OK
```

followed by the GNSS data:

```
⇐ <GNSS data>
```

If the client already knows beforehand that a mountpoint is password protected, it can send the password with the first request message.

Example:

```
⇒ GET /BUCU1 HTTP/1.0
  User-Agent: NTRIP GNSSInternetRadio/1.2.0
  Authorization: Basic aHVnb2JlbjpodWdvYmVuMTIz
```

The NtripCaster will again answer

```
⇐ ICY 200 OK
```

followed by the GNSS data:

```
⇐ <GNSS data>
```

The client password is different from the server password. The client password is coded like the HTTP Basic Authentication Scheme [4] and allows the client access to protected content. The "basic" authentication scheme is based on the model that the user agent must authenticate with a user-ID and a password for each realm (here mountpoint). The realm value should be considered as an opaque string that can only be compared for equality with other realms on that server. The server will authorize the request only if it can validate the user-ID and password for the protected space of the requested mountpoint. There are no optional authentication parameters. To receive

authorization, the client sends the user-ID and password, separated by a single colon (":") character and within a “base64” [5] encoded string in the credentials. If the user agent wishes to send the user-ID "Aladdin" and password "open sesame", it would use the following header field:

```
⇒ Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

The basic authentication scheme is a non-secure method of filtering unauthorized access to resources on an HTTP server. It is based on the assumption that the connection between the client and the server can be regarded as a trusted carrier. As this is not generally true on an open network, the basic authentication scheme should be used accordingly. In spite of this, clients should implement the scheme in order to communicate with servers that use it [4].

5.2 DIGEST AUTHENTICATION SCHEME

For applications that require more security with user authentication, the Digest Access Authentication for HTTP specified in RFC 2617 [6] can be used as an alternative.

Like Basic, Digest access authentication verifies that both communicating parties know a shared secret (a password). Unlike Basic, this verification can be done without sending the password in the clear, which is Basic's biggest weakness.

Like Basic Access Authentication, the Digest scheme is based on a simple challenge-response paradigm. The Digest scheme challenges using a nonce value. A valid response contains a checksum (for Ntrip the MD5 [7] checksum is compulsory) of the username, the password, the given nonce value, the HTTP method, and the requested URI. Thus, the password is never sent in the clear.

The Digest Access Authentication scheme is conceptually similar to the Basic scheme. The format of the modified WWW-Authenticate header line and the Authorization header line are specified in RFC 2617. [6]

5.3 NMEA REQUEST MESSAGES

For some network-dependent applications it is necessary to send the position of the NtripClient to the NtripCaster. That position could be used by the NtripCaster to provide a data stream for a Virtual Reference Station (VRS) or to determine the best data stream to broadcast. Ntrip allows clients to send NMEA strings after the HTTP request for a data stream.

If the <nmea> parameter is set to “1” (see Source-table section), the NtripCaster must receive at least one NMEA GGA string to prepare the data and start sending. The NtripClient is allowed to send more than one NMEA GGA string or NMEA strings of other type than GGA at any time.

The following is an example of a request to a source-table named “vrs_bayern” to get a data stream generated for a virtual reference station with the coordinates of the NMEA string:

⇒ GET /vrs_bayern HTTP/1.1<CR><LF>

Accept: rtk/rtcm, dgps/rtcm<CR><LF>

User-Agent: NTRIP Survey-Controller-15.0<CR><LF>

<CR><LF>

$GPGGA,165631.00,4810.8483085,N,01139.900759,E,1,05,01.9,+00400,M,,M,,*??<CR><LF>

Note that sending an NMEA string containing latitude and longitude information allows an NtripCaster to track the NtripClient's position. The operator of an NtripCaster may wish to consider informing clients about this potential privacy problem.

This page intentionally left blank.

6 SOURCE-TABLE

The NtripCaster maintains a source-table containing information on available NtripSources, networks of NtripSources, and NtripCasters, to be sent to an NtripClient on request. Note that to request a source-table from the NtripCaster, the NtripClient uses the client message (see Fig. 3) while leaving out the mountpoint parameter.

Source-table records are dedicated to one of the following:

a) Data STReams (record type STR),
b) CASters (record type CAS), or
c) NETworks of data streams (record type NET).

This structure is expandable by introducing new record types when necessary. Older NtripClient versions might ignore newly introduced record types. All NtripClients must be able to decode record type STR. Decoding types CAS and NET is an optional feature.

All data fields in the source-table records are separated using the semicolon character ";" as a field delimiter. In case the semicolon character becomes part of the content of a data field, it must be quoted: ";". The number of data fields in the source-table records is not fixed. Introducing additional data fields is allowed. The last data field always contains "Miscellaneous" information.

The source-table is initiated by the HTTP/1.1 header fields

Server: <NtripCasterIdentifier>/<NtripVersion><CR><LF>
Content-Type: text/plain<CR><LF>
Content-Length: <Content-Length><CR><LF>
<CR><LF>

followed by the actual source-table records. The "Server:" record contains

- Before the slash: an NtripCaster identifier to be defined by the NtripCaster operator (see parameter #4 in Table 2)
- After the slash: an Ntrip version number (e.g. NtripV1.0 or 1.0) in order to allow an NtripClient to understand which version of Ntrip is supported by a particular NtripCaster.

The content-length indicates the size of the source-table records (decimal number of octets, e.g. "Content-Length: 243").

The end of the source-table is notified by the string: ENDSOURCETABLE

Table 1: Format and Contents of Source-Table Records Describing a Data Stream

#	Record Parameter	Meaning	Format	Examples
1	<type> = STR	The following parameters of this record describe a data stream	3 Characters	STR (the only acceptable string)
2	<mountpoint>	Caster mountpoint	Characters, 100 max.	LEIJ0 LEIJ1 WTZJ0

Table 1: Format and Contents of Source-Table Records Describing a Data Stream

#	Record Parameter	Meaning	Format	Examples
3	<identifier>	Source identifier, e.g. name of city next to source location	Characters, undefined length	Frankfurt
4	<format>	Data format RTCM, RAW, etc.	Characters, undefined length	RTCM 2 RTCM 2.0 RTCM 2.1 RTCM 2.3 RTCM 3 RTCM SAPOS CMR CMR+ RAW RTCA
5	<format-details>	E.g. RTCM message types or RAW data format etc., update periods in parenthesis in seconds	Characters, undefined length	1(1), 2(1), 3(30) MBEN(1) LB2
6	<carrier>	Data stream contains carrier phase information 0 = No (e.g. for DGPS) 1 = Yes, L1 (e.g. for RTK) 2 = Yes, L1&L2 (e.g. for RTK)	Integer	0 1 2
7	<nav-system>	Navigation system(s)	Characters, undefined length	GPS GPS+GLONASS GPS+EGNOS
8	<network>	Network	Characters, undefined length	EUREF IGS IGLOS SAPOS GREF Misc
9	<country>	Three character country code in ISO 3166	3 Characters	DEU ITA ESP
10	<latitude>	Position, latitude, north (approximate position in case of nmea = 1)	Floating point number, two digits after decimal point	40.12 -12.14
11	<longitude>	Position, longitude, east (approximate position in case of nmea = 1)	Floating point number, two digits after decimal point	10.12 357.85
12	<nmea>	Necessity for Client to send NMEA message with approximate position to Caster 0 = Client must not send NMEA message with approximate position to Caster 1 = Client must send NMEA GGA message with approximate position to Caster	Integer	0 1
13	<solution>	Stream generated from single reference station or from networked reference stations 0 = Single base 1 = Network	Integer	0 1

Table 1: Format and Contents of Source-Table Records Describing a Data Stream

#	Record Parameter	Meaning	Format	Examples
14	<generator>	Hard- or software generating data stream	Characters, undefined length	JPS Legacy E GPSNet
15	<compr-encryp>	Compression/Encryption algorithm applied	Characters, undefined length	none
16	<authentication>	Access protection for this particular data stream N = None B = Basic D = Digest	1 Character	N B D
17	<fee>	User fee for receiving this particular data stream N = No user fee Y = Usage is charged	1 Character	N Y
18	<bitrate>	Bit rate of data stream, bits per second	Integer	500 5000
…	…			
…	…			
n	<misc>	Miscellaneous information, last data field in record	Characters, undefined length	none Demo

Table 2: Format and Contents of Source-Table Records Describing a Caster

#	Source-table Element	Meaning	Format	Examples
1	<type> = CAS	The following parameters of this record describe a Caster	3 Characters	CAS (the only acceptable string)
2	<host>	Caster Internet host domain name or IP address	Characters, 128 max.	141.74.243.11 euref-ip.ifag.de
3	<port>	Port number	Integer	80 2101
4	<identifier>	Caster identifier, e.g. name of provider	Characters, undefined length	NTRIP Caster/0.5.3 Trimble-iGate
5	<operator>	Name of institution / agency / company operating the Caster	Characters, undefined length	BKG Geo++
6	<nmea>	Capability of Caster to receive NMEA message with approximate position from Client 0 = Caster is not able to handle incoming NMEA message with approximate position from Client 1 = Caster is able to handle incoming NMEA GGA message with approximate position from Client	Integer	0 1
7	<country>	Three character country code in ISO 3166	3 Characters	DEU ITA ESP

Table 2: Format and Contents of Source-Table Records Describing a Caster

#	Source-table Element	Meaning	Format	Examples
8	<latitude>	Position, latitude, north	Floating point number, two digits after decimal point	40.12 -12.14
9	<longitude>	Position, longitude, east	Floating point number, two digits after decimal point	10.12 357.85
10	<fallback_host>	Fallback Caster IP address No Fallback: 0.0.0.0	Characters, 128 max.	213.20.169.236 caster.fgi.fi 0.0.0.0
11	<fallback_port>	Fallback Caster port number No Fallback: 0	Integer	80 2101 0
…				
…				
n	<misc>	Miscellaneous information, last data field in record	Characters, undefined length	none

Table 3: Format and Contents of Source-Table Records Describing a Network of Data Streams

#	Source-table Element	Meaning	Format	Examples
1	<type> = NET	The following parameters of this record describe a network of data streams	3 Characters	NET (the only acceptable string)
2	<identifier>	Network identifier, e.g. name of a network of GNSS permanent reference stations	Characters, undefined length	EPN IGLOS GREF SAPOS-THR
3	<operator>	Name of institution / agency / company operating the network	Characters, undefined length	EUREF IGS TRIMBLE ASCOS GEO++ SAPOS-THR
4	<authentication>	Access protection for data streams of the network N = None B = Basic D = Digest	1 Character	N B D N,B
5	<fee>	User fee for receiving data streams from this network N = No user fee Y = Usage is charged	1 Character	N Y N,Y
6	<web-net>	Web-address for network information	Characters, undefined length	http://igs.ifag.de
7	<web-str>	Web-address for stream information	Characters, undefined length	http://www.epncb.oma.be none

Table 3: Format and Contents of Source-Table Records Describing a Network of Data Streams

#	Source-table Element	Meaning	Format	Examples
8	<web-reg>	Web address or mail address for registration	Characters, undefined length	euref-ip@ifag.de http://igs.ifag.de
…				
…				
n	<misc>	Miscellaneous information, last data field in record	Characters, undefined length	none

This page intentionally left blank.

7 REFERENCES

[1] Gebhard, H., and R. Kays: "*Real-Time Streaming of Differential GPS Corrections via Internet*", Feasibility Study, Informatik Centrum Dortmund, Germany, unpublished, March 2002

[2] Weber, G.: "*Echtzeit-Übertragung von RTCM-Daten über Internet und Mobilfunk*", Beitrag zum 57. DVW-Seminar "GPS 2002: Antennen, Höhenbestimmung und RTK Anwendungen", 16.-17. Sep 2002, Karlsruhe, Schriftenreihe Deutscher Verein für Vermessungswesen, Band 44, S. 107-116, Stuttgart, 2002

[3] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee: "*Hypertext Transfer Protocol -- HTTP/1.1*", RFC 2616, 1999

[4] Berners-Lee, T., Fielding, R., and H. Frystyk: "*Hypertext Transfer Protocol -- HTTP/1.0*", RFC 1945, 1996

[5] Freed, N., and N. Borenstein: "*Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*", RFC 2045, 1996

[6] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart: "*HTTP Authentication: Basic and Digest Access Authentication*", RFC 2617, 1999

[7] Rivest, R., "*The MD5 Message-Digest Algorithm*", RFC 1321, 1992

This page intentionally left blank.

APPENDIX A

EXAMPLE SOURCE-TABLE

SOURCETABLE 200 OK
Server : NTRIP Caster 1.5.5/1.0
Date : 23/Jan/2004:08:54:59 UTC
Content-Type: text/plain
Content-Length: 6999

CAS;129.217.182.51;80;EUREF;BKG;0;DEU;51.5;7.5;http://igs.ifag.de/index_ntrip_cast.htm
CAS;62.159.109.248;8080;Trimble GPSNet;Trimble Terrasat;1;DEU;48.03;11.72;http://www.virtualrtk.com
NET;EUREF;EUREF;B;N;http://www.epncb.oma.be/euref_IP;http://www.epncb.oma.be/euref_IP;http://www.epncb.oma.be/projects/euref_IP/NtripRegister.html;none
NET;IGS;BKG;B;N;http://igscb.jpl.nasa.gov/;none;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;GREF;BKG;B;N;http://igs.ifag.de/gref_realtime.htm;http://igs.ifag.de/root_ftp/misc/ntrip/streamlist.htm;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;ascos;Ruhrgas AG;B;N;http://www.ascos.de;none;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;SCIGN-OCRT;SOPAC,CSRC;B;N;http://www.scign.org;none;ybock@ucsd.edu;none
NET;CORS;NGS;B;N;http://www.ngs.noaa.gov/CORS;http://igs.ifag.de/root_ftp/misc/ntrip/streamlist.htm;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;Misc;BKG;B;N;http://igs.ifag.de/root_ftp/misc/ntrip/streamlist.htm;none;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;EGNOS;BKG;B;N;http://igs.ifag.de/root_ftp/misc/ntrip/streamlist.htm;none;http://igs.ifag.de/root_ftp/software/NtripRegister.doc;none
NET;OmniSTAR;BKG;B;Y;http://www.omnistar.nl;http://www.omnistar.nl/graphs.asp;http://www.omnistar.nl/purchase.asp;Test and Evaluation only
STR;FFMJ2;Frankfurt;RTCM 2.1;1(1),3(19),16(59);0;GPS;GREF;DEU;50.12;8.68;0;1;GPSNet V2.10;none;N;N;560;Demo
STR;FFMJ1;Frankfurt;RTCM 2.2;3(19),16(59),18(1),19(1);2;GPS;GREF;DEU;50.09;8.66;0;0;GPSNet V1.9;none;N;N;2800;Demo
STR;FFMT0;Frankfurt;RTCM 2.0;1(3),2(90),3(90),16(90);0;GPS;Misc;DEU;50.09;8.66;0;0;Trimble 4000SSI;none;N;N;190;ALF Demo
STR;CONZ0;Concepcion-TIGO;RAW;Compact(1);2;GPS+GLO;IGS;CHL;-36.84;286.98;0;0;Javad Legacy E;none;B;N;3600;none
STR;FFMJ0;Frankfurt;RAW;Compact(1);2;GPS+GLO;IGS;DEU;50.09;8.66;0;0;Javad Legacy E;none;N;N;3600;Demo
STR;LEIJ0;Leipzig;RAW;Compact(1);2;GPS+GLO;IGS;DEU;51.33;12.37;0;0;Javad Legacy E;none;B;N;3600;none
STR;LDBG0;Lindenberg;RAW;Compact(1);2;GPS+GLO;IGS;DEU;52.21;14.12;0;0;Javad Legacy E;none;B;N;3600;none
STR;MOXA0;Moxa;RAW;Compact(1);2;GPS+GLO;IGS;DEU;50.64;11.61;0;0;Javad Legacy E;none;B;N;3600;none
STR;MUEJ0;Muenchen;RAW;Compact(1);2;GPS+GLO;IGS;DEU;48.13;11.57;0;0;Javad Legacy E;none;B;N;3600;none
STR;WARN0;Warnemuende;RAW;Compact(1);2;GPS+GLO;IGS;DEU;54.17;12.10;0;0;Javad Legacy E;none;B;N;3600;none
STR;BARC0;Barcelona;RTCM 2.0;1(1),3(3),16(90);0;GPS;Misc;ESP;41.37;2.16;0;0;Novatel OEM3;none;B;N;230;ICC, Geod. Dep.
STR;BOCH0;Bochum;RTCM 2.1;1(1)3,16,18(1),19(1);2;GPS;Misc;DEU;50.45;7.27;0;0;none;none;B;N;13000;FH Bochum
STR;BRUS0;Brussels;RTCM 2.0;1(1),3(60),16;0;GPS;Misc;BEL;50.80;4.36;0;0;Ashtech UZ-12;none;B;N;500;ROB
STR;SP300;Ephemerides;SP3;ASCII, last 8 15min Epochs;0;GPS;Misc;FIN;60.10;24.50;0;1;IGS Ultra Rapid Orbit;none;B;N;5600;Finnish Geodetic Institute
STR;BUCU0;Bucharest;RTCM 2.0;1(1),3(60),16(60);0;GPS;EUREF;ROU;44.46;26.12;0;0;Ashtech Z-XII3;none;B;N;520;TU Bucharest
STR;CAGZ0;Cagliari;RTCM 2.1;1(3),3(60),16(60),18(1),19(1),31(3);2;GPS+GLO;EUREF;ITA;39.14;8.97;0;0;JPS Eurocard;none;B;N;3900;Univ Cagliari, DIST
STR;STEI0;Steinbrink;RAW;Compact(1);2;GPS+GLO;ascos;DEU;52.50;8.73;0;0;Javad Legacy E;none;B;N;3600;ALLSAT
STR;UMME0;Ummeln;RAW;Compact(1);2;GPS+GLO;ascos;DEU;51.98;8.46;0;0;Javad Legacy E;none;B;N;3600;ALLSAT
STR;HELS0;Helsinki;RTCA;Thales ASCII;1;EGNOS;Misc;FIN;60.10;24.50;0;0;Thales DG-16;none;B;N;700;Finnish Geodetic Institute
STR;HELS1;Helsinki;RTCM 3.0;1,2,3,6,9,16;0;GPS,EGNOS;Misc;FIN;60.10;24.50;0;0;Thales DG-16;none;B;N;700;Finnish Geodetic Institute
STR;HERT0;Hailsham;RTCM 2.2;1(1),3(60),18(1),19(1),22(60),31(1);2;GPS+GLO;EUREF;GBR;50.87;0.33;0;0;Ashtech Z18;none;B;N;5200;NERC
STR;KRAW0;Krakow;RTCM 2.2;1(1),3(60),16(60),18(1),19(1),22(60);2;GPS;EUREF;POL;50.01;19.92;0;0;Ashtech UZ-12;none;B;N;1900;AGH
STR;KOPE0;Koper;RTCM 2.1;1(5),3(69),20(1),21(1),22;2;GPS;Misc;SLO;48.40;17.03;0;0;none;none;B;N;2500;Harpha Sea
STR;GAVL0;Gavle;RTCM 2.3;1(1),3(19),16(59);0;GPS+GLO;Misc;SWE;60.67;17.14;0;0;Javad Legacy;none;B;N;800;National Land Survey Sweden
STR;MILU0;Milano;RTCM 2.1;1(1),3(60),16(60),18(1),19(1);2;GPS;Misc;ITA;45.48;9.23;0;0;Ashtech Z12;none;B;N;2800;University Milano, Agricultural Institute
STR;MADR0;Madrid;RTCM 2.1;1(1),3(10),16,18(1),19(1);2;GPS;Misc;ESP;40.44;356.29;0;0;Ashtech ZSensor;none;B;N;3300;IGNE, Serv. Geod.
STR;MALA0;Malaga;RTCM 2.0;1(1),2(1),3(30);0;GPS;Misc;ESP;36.73;355.61;0;0;Trimble 4000SSI;none;B;N;1200;IGNE, Serv. Geod.
STR;OBET0;Oberpfaffenhofen;RAW;MBEN/PBEN/SNAV/SALM(1);2;GPS;Misc;DEU;48.08;11.28;0;0;Ashtech Z12;none;B;N;8000;DLR
STR;PADO0;Padova;RTCM 2.1;1(1),3(10),16(30),18(1),19(1),59;2;GPS;IGS;ITA;45.41;11.90;0;0;Trimble 4000SSI;none;B;N;3600;Univ Padova
STR;MAT10;Matera;RTCM 2.1;3(60),16(60),18(1),19(1),22(60),36(1);2;GPS+GLO;IGS;ITA;40.65;16.70;0;0;Ashtech Z18;none;B;N;5600;Agenzia Spaziale Italiana
STR;GOPE0;Praha-Ondrejov;RTCM 2.2;1(1),3(60),16(60),18(1),19(1),22(60),31(1),36(1);2;GPS+GLO;EUREF;CZE;49.91;14.79;0;0;Ashtech Z18;none;B;N;5600;Uni Praha
STR;ROMA0;Roma;RTCM 2.1;1(1),3(10),18(1),19(1),59;2;GPS;Misc;ITA;42.00;12.50;0;0;Trimble 4000;none;B;N;3400;Univ Roma
STR;ROMG0;Roma;RTCM 2.1;9(1),34(1);0;GPS;Misc;ITA;42.00;12.50;0;0;Trimble 5700;none;B;N;500;Galileo Sistemi
STR;SYDN0;Sydney;RTCM 2.1;3(30),18(1),19(1);2;GPS;Misc;AUS;-33.92;151.23;0;0;Leica MC500;none;B;N;3200;University of New South Wales
STR;TORI0;Torino;RTCM 2.1;1(1),3(60),16(30),18(1),19(1);2;GPS;Misc;ITA;45.06;7.66;0;0;none;none;B;N;12000;Politecnico di Torino
STR;PENC0;Penc-RT;RTCM 2.0;1(1),3(10),22,59;0;GPS;EUREF;HUN;47.79;19.28;0;0;Trimble 5700;none;B;N;700;FOMI, SGO
STR;SZEK0;Szekesfehervar;RTCM 2.0;1(1),3(10);0;GPS;Misc;HUN;47.18;19.42;0;0;GPSBase;none;B;N;700;FOMI, SGO
STR;BERJ0;Berlin;RTCM 2.1;1(1),3(19),16(59);0;GPS;GREF;DEU;52.52;13.38;0;1;GPSNet V2.10;none;B;N;560;VRS
STR;DREJ1;Dresden;RTCM 2.1;1(1),3(19),16(59);0;GPS;GREF;DEU;51.05;13.73;0;1;GPSNet V2.10;none;B;N;560;VRS
STR;WILH0;Wilhelmshaven;RTCM 2.1;1(1),3(19),16(59);0;GPS;GREF;DEU;53.52;8.10;0;1;GPSNet V2.01;none;B;N;560;VRS
STR;LEID0;Leidschendam;RTCM 2.0;1(1);0;GPS;OmniSTAR;NLD;52.10;4.41;0;1;OmniSTAR;none;B;Y;120;EA-SAT Geostationary L-band Broadcast
STR;ERLA0;Erlanger;RTCM 2.2;3(60),18(1),19(1),22(60);2;GPS;CORS;USA;39.01;275.24;0;0;Ashtech Z-XII3;none;B;N;3000;http://ncad.net
STR;SBCC0;Mission-Viejo;RTCM 2.1;3(15),18(1),19(1),22(15),59(1);2;GPS;SCIGN-OCRT;USA-CA;33.55;242.34;0;0;Ashtech Z-XII3;none;B;N;4000;none
STR;SACY0;Santa-Ana;RTCM 2.1;3(15),18(1),19(1),22(15),59(1);2;GPS;SCIGN-OCRT;USA-CA;33.74;242.10;0;0;Ashtech Z-XII3;none;B;N;4000;none
STR;SBCC1;Mission-Viejo;RAW;MBEN(1);2;GPS;SCIGN-OCRT;USA-CA;33.55;242.34;0;0;Ashtech Z-XII3;none;B;N;7300;none
STR;SACY1;Santa-Ana;RAW;MBEN(1);2;GPS;SCIGN-OCRT;USA-CA;33.74;242.10;0;0;Ashtech Z-XII3;none;B;N;7300;none
STR;TIGO0;TIGO-TEST;RAW;PBEN(5),MBEN(5);2;GPS;Test;CHL;-36.84;286.98;0;0;Ashtech Z12;none;B;N;1500;Softwaretest
ENDSOURCETABLE

This page intentionally left blank.

APPENDIX B

Format Specifications

The following provides additional information on format specifications as introduced through parameters <format> and <format-details> in the source-table (see Section 6, Source-Table). Currently a number of RTCM formats for differential GNSS data and raw GNSS data formats are supported along with the formats RTCA, RINEX, BINEX, and SP3.

Each source-table record of type "STR" contains keywords to describe the data format and format details provided in a specific data stream. The following are example definition for these parameters. Introducing provider-dependent other format keywords and format details is allowed.

B.1. RTCM Formats

Differential GNSS corrections are often available in RTCM format. Table 1 defines RTCM keywords to be used as <format> parameter in source-table records of type STR and the RTCM messages corresponding to these keywords as <format-details>. Providing a Minimum Message Set (MMS) is mandatory when mentioned. A provider may add more messages. The update rate (seconds) is not defined but may be given in brackets after each message type.

Table B-1: RTCM format and Format-Details Description

Format / Keyword	Comment	Format Details / Messages
RTCM 2.0	DGPS	MMS: 1, 3
RTCM 2.1	RTK	MMS: 3, 18, 19
RTCM 2.3	RTK	MMS: 18, 19, 23, 24
RTCM 2	Messages based on fixed status of RTCM 2.x document. If fixed status has not been achieved, messages based on last tentative description as officially documented	1, 3, 18, 19, 20, 21,23, 24
RTCM 3	Messages based on primary document of RTCM Version 3	1004, 1007, 1008, 1009
RTCM SAPOS	Subset of RTCM messages defined as SAPOS standard, message 59 contains FKP information, see http://www.geopp.de/download/geopp-rtcm-fkp59.pdf	20, 21, 23, 24, 59

B.2. Other Formats

Most GNSS equipment vendors have their own proprietary formats. These "raw" GNSS data may be disseminated using Ntrip. Furthermore, Ntrip may also disseminate data in other formats. All such uses of Ntrip are beyond the scope of this document, and are not to be considered as part of the Ntrip standard. Table 2 shows examples for source-table parameters <format> and <format-details> when streaming such data.

Table B-2: Format and Format Details Descriptions Beside RTCM, Examples

Format / Keyword	Comment	Format Details / Messages
CMR	Compact Measurement Record (CMR), publicly documented format, accepted by several GNSS equipment vendors, among those providing some interoperability across vendors	none
CMR+	Proprietary format	none
SAPOS-AdV	Proprietary format	none
RTCA	Radio Technical Commission, Aviation	0, 2
RAW	Raw observation data from GNSS receiver	Examples: - RT17 (Trimble) - Concise (Trimble) - MBEN (Ashtech) - LB2 (Leica) - Compact (Topcon/Javad)
RINEX	Receiver Independent Exchange Format (RINEX) for GNSS observation data, see http://igscb.jpl.nasa.gov/igscb/data/format/rinex2.txt	none
SP3	Standard Product #3 (SP3), GNSS orbit data, see http://www.ngs.noaa.gov/GPS/SP3_format.html	none
BINEX	Binary Exchange (BINEX) format for GPS, GLONASS, SBAS data, metadata, ephemeredes, orbits or solutions, see http://binex.unavco.org	none

APPENDIX C

Example Client Source Code

```
/*
Easy example NTRIP client for Linux/Unix.
Copyright (C) 2003 by Dirk Stoecker <dirk.stoecker@euronav.de >

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public
License as published by the Free Software Foundation; either version 2 of the License, or  (at your option) any later
version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA or read
http://www.gnu.org/licenses/gpl.txt
*/

/* Version history
  Please always keep revision history and the two related strings up to date!
  1.1   2003-02-24 stoecker    initial version
  1.2   2003-02-25 stoecker    fixed agent string
*/

#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>

/* The string, which is sent as agent in HTTP request */
#define AGENTSTRING "NTRIP LinuxClient"

#define MAXDATASIZE 1000 /* max number of bytes we can get at once */
char buf[MAXDATASIZE];

/* CVS revision and version */
static char revisionstr[] = "$Revision: 1.3 $";
static char datestr[]     = "$Date: 2003/03/20 13:00:00 $";

struct Args
{
  char *server;
  int   port;
  char *user;
  char *password;
  char *data;
};
```

```
/* option parsing */
#ifdef NO_LONG_OPTS
#define LONG_OPT(a)
#else
#define LONG_OPT(a) a
static struct option opts[] = {
{ "data",       required_argument, 0, 'd'},
{ "server",     required_argument, 0, 's'},
{ "password",   required_argument, 0, 'p'},
{ "port",       required_argument, 0, 'r'},
{ "user",       required_argument, 0, 'u'},
{ "help",       no_argument,       0, 'h'},
{0,0,0,0}};
#endif
#define ARGOPT "d:hp:r:s:u:"

static int getargs(int argc, char **argv, struct Args *args)
{
  int res = 1;
  int getoptr;
  char *a;
  int i = 0, help = 0;
  char *t;

  args->server = "129.217.182.51";
  args->port = 80;
  args->user = "";
  args->password = "";
  args->data = 0;
  help = 0;

  do
  {
#ifdef NO_LONG_OPTS
    switch((getoptr = getopt(argc, argv, ARGOPT)))
#else
    switch((getoptr = getopt_long(argc, argv, ARGOPT, opts, 0)))
#endif
    {
    case 's': args->server = optarg; break;
    case 'u': args->user = optarg; break;
    case 'p': args->password = optarg; break;
    case 'd': args->data = optarg; break;
    case 'h': help=1; break;
    case 'r':
      args->port = strtoul(optarg, &t, 10);
      if((t && *t) || args->port < 1 || args->port > 65535)
        res = 0;
      break;
    case -1: break;
    }
  } while(getoptr != -1 || !res);

  for(a = revisionstr+11; *a && *a != ' '; ++a)
    revisionstr[i++] = *a;
  revisionstr[i] = 0;
  datestr[0] = datestr[7];
  datestr[1] = datestr[8];
```

```
  datestr[2] = datestr[9];
  datestr[3] = datestr[10];
  datestr[5] = datestr[12];
  datestr[6] = datestr[13];
  datestr[8] = datestr[15];
  datestr[9] = datestr[16];
  datestr[4] = datestr[7] = '-';
  datestr[10] = 0;

  if(!res || help)
  {
    fprintf(stderr, "Version %s (%s) GPL\nUsage: %s -s server -u user ...\n"
    " -d " LONG_OPT("--data     ") "the requested data set\n"
    " -s " LONG_OPT("--server   ") "the server name or address\n"
    " -p " LONG_OPT("--password ") "the login password\n"
    " -r " LONG_OPT("--port     ") "the server port number (default 80)\n"
    " -u " LONG_OPT("--user     ") "the user name\n"
    , revisionstr, datestr, argv[0]);
    exit(1);
  }
  return res;
}

static char encodingTable [64] = {
  'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P',
  'Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f',
  'g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v',
  'w','x','y','z','0','1','2','3','4','5','6','7','8','9','+','/'
};

/* does not buffer overrun, but breaks directly after an error */
/* returns the number of required bytes */
static int encode(char *buf, int size, char *user, char *pwd)
{
  unsigned char inbuf[3];
  char *out = buf;
  int i, sep = 0, fill = 0, bytes = 0;

  while(*user || *pwd)
  {
    i = 0;
    while(i < 3 && *user) inbuf[i++] = *(user++);
    if(i < 3 && !sep)    {inbuf[i++] = ':'; ++sep; }
    while(i < 3 && *pwd)  inbuf[i++] = *(pwd++);
    while(i < 3)         {inbuf[i++] = 0; ++fill; }
    if(out-buf < size-1)
      *(out++) = encodingTable[(inbuf [0] & 0xFC) >> 2];
    if(out-buf < size-1)
      *(out++) = encodingTable[((inbuf [0] & 0x03) << 4)
              | ((inbuf [1] & 0xF0) >> 4)];
    if(out-buf < size-1)
    {
      if(fill == 2)
        *(out++) = '=';
      else
        *(out++) = encodingTable[((inbuf [1] & 0x0F) << 2)
                | ((inbuf [2] & 0xC0) >> 6)];
    }
```

```
    if(out-buf < size-1)
    {
     if(fill >= 1)
       *(out++) = '=';
     else
       *(out++) = encodingTable[inbuf [2] & 0x3F];
    }
    bytes += 4;
  }
  if(out-buf < size)
    *out = 0;
  return bytes;
}

int main(int argc, char **argv)
{
  struct Args args;

  if(getargs(argc, argv, &args))
  {
    int i, sockfd, numbytes;
    char buf[MAXDATASIZE];
    struct hostent *he;
    struct sockaddr_in their_addr; /* connector's address information */

    if(!(he=gethostbyname(args.server)))
    {
     perror("gethostbyname");
     exit(1);
    }
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
    {
     perror("socket");
     exit(1);
    }
    their_addr.sin_family = AF_INET;    /* host byte order */
    their_addr.sin_port = htons(args.port);  /* short, network byte order */
    their_addr.sin_addr = *((struct in_addr *)he->h_addr);
    memset(&(their_addr.sin_zero), '\0', 8);
    if(connect(sockfd, (struct sockaddr *)&their_addr,
    sizeof(struct sockaddr)) == -1)
    {
     perror("connect");
     exit(1);
    }

    if(!args.data)
    {
     i = snprintf(buf, MAXDATASIZE,
     "GET / HTTP/1.1\r\n"
     "User-Agent: %s/%s\r\n"
//     "Accept: */*\r\n"
//     "Connection: close\r\n"
     "\r\n"
     , AGENTSTRING, revisionstr);
    }
    else
    {
```

```
    i=snprintf(buf, MAXDATASIZE-40, /* leave some space for login */
    "GET /%s HTTP/1.1\r\n"
    "User-Agent: NTRIP NtripLinuxClient1.3 \r\n"
//    "Accept: */*\r\n"
//    "Connection: close\r\n"
    "Authorization: Basic "
    , args.data, AGENTSTRING, revisionstr);
    if(i > MAXDATASIZE-40 && i < 0) /* second check for old glibc */
    {
     fprintf(stderr, "Requested data too long\n");
     exit(1);
    }
    i += encode(buf+i, MAXDATASIZE-i-5, args.user, args.password);
    if(i > MAXDATASIZE-5)
    {
     fprintf(stderr, "Username and/or password too long\n");
     exit(1);
    }
    snprintf(buf+i, 5, "\r\n\r\n");
    i += 5;
   }
   if(send(sockfd, buf, i, 0) != i)
   {
    perror("send");
    exit(1);
   }
   if(args.data)
   {
    int k = 0;
    while((numbytes=recv(sockfd, buf, MAXDATASIZE-1, 0)) != -1)
    {
     if(!k)
     {
      if(numbytes != 12 || strncmp("ICY 200 OK\r\n", buf, 12))
      {
       fprintf(stderr, "Could not get the requested data\n");
       exit(1);
      }
      ++k;
     }
     else
      fwrite(buf, numbytes, 1, stdout);
    }
   }
   else
   {
    while((numbytes=recv(sockfd, buf, MAXDATASIZE-1, 0)) != -1)
    {
     fwrite(buf, numbytes, 1, stdout);
     if(!strncmp("ENDSOURCETABLE\r\n", buf+numbytes-16, 16))
      break;
    }
   }

   close(sockfd);
  }
  return 0;
}
```

This page intentionally left blank.