Networked Transport of RTCM via Internet Protocol

Ntrip, Version 1.0

Design – Protocol – Software

Part I & II

Developed within the framework of EUREF, European Sub-Commission of Commission X (Global and Regional Geodetic Networks) of the International Association of Geodesy (IAG)

Published by Federal Agency for Cartography and Geodesy (BKG), Frankfurt, Germany

http://igs.ifag.de/index_ntrip.htm euref-ip@bkg.bund.de
Authors
Harald Gebhard
Georg Weber

Contributions from
Rüdiger Kays
Denise Dettmering
Christian Pagels (Part I)
Sebastian Pötschke (Part I)
Ken Fortune (Part I)
Dirk Stöcker (Linux Client Example)

Affiliations
1 Lehrstuhl für Kommunikationstechnik, Universität Dortmund, Germany
2 Bundesamt für Kartographie und Geodäsie (BKG), Frankfurt, Germany
3 Trimble Terrasat GmbH, Höhenkirchen, Germany
4 Trimble New Zealand Limited, Christchurch, New Zealand
5 Geodätisches Institut, TU Dresden, Germany
General Note

Networked Transport of RTCM via Internet Protocol (Ntrip) stands for an application-level protocol 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 enhanced to GNSS data streams.

Ntrip is designed for disseminating differential correction data (e.g. in the RTCM-104 format) 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 is implemented in three system software components: NtripClients, NtripServers and NtripCasters. The NtripCaster is the actual HTTP server program whereas NtripClient and NtripServer are acting as HTTP clients.

Ntrip is meant to be an open none-proprietary protocol. Major characteristics of Ntrip’s dissemination technique are:

- Based on the popular HTTP streaming standard; comparatively easy to implement when having limited client and server platform resources available.
- Application not limited to one particular plain or coded stream content; ability to distribute any kind of GNSS data.
- Potential to support mass usage; disseminating hundreds of streams simultaneously for up to thousand users possible when applying modified Internet Radio broadcasting software.
- Considering security needs; stream providers and users don’t necessarily get into contact, streams often not blocked by firewalls or proxyservers protecting Local Area Networks.
- Enables streaming over any mobile IP network because of using TCP/IP.

This Ntrip documentation will provide information to readers with diverse levels of IT knowledge. Its information is therefore given in a more narrative language, trying to avoid a less comprehensible technical diction.

This documentation describes Ntrip Version 1.0. The work done so far for Part I sections “Client Messages” and “Source-Table” has been frozen on March 3rd, 2003 in order to allow the development of Ntrip-based implementations and applications on a firm and fully anchored basis. Further Ntrip versions will be issued when necessary. Contributions from anyone to the ongoing work on Ntrip are highly appreciated.
Document History

15 Jan 2003  Draft
24 Jan 2003  Published via Internet
28 Jan 2003  Wording, format, stylistics
30 Jan 2003  Source-table edited, parameter <vrs> renamed to <nmea>
31 Jan 2003  NMEA Request Messages included
03 Feb 2003  Modification of server and client messages, screenshots updated, Downloads section included
04 Feb 2003  NtripCaster hardware configuration example and proxyserver option included
05 Feb 2003  Additional explanations in Example Implementation section
10 Feb 2003  Appendix B extended, meaning of parameter <protection> in source-table changed
20 Feb 2003  Source-table parameters <solution>, <carrier>, <authentication>, and <fee> added, parameter <protection> deleted, Appendix B split into two sections
25 Feb 2003  Wording corrected
26 Feb 2003  Appendix A, new example source-table included
27 Feb 2003  Source-table: parameter <operator> included in CAS-records, sequence of parameters in STR-records changed, meaning of parameter <carrier> changed
03 Mar 2003  Status of work on technical content of Part I frozen as “Ntrip Version 1.0”
25 Mar 2003  Appendix C, Linux Client Example added
04 Apr 2003  GNSS Internet Radio screenshots updated
19 May 2003  GNSS Internet Radio screenshots updated
18 June 2003  Co-Processor included in Broadcaster hardware recommendation section
20 June 2003  Source-Agent included in Server Message section, Downloads section extended
# Table of Contents

## PART I  Networked Transport of RTCM via Internet Protocol ............................. 6
1. Introduction ......................................................................................................................... 7
2. System Concept .................................................................................................................. 8
3. System Elements .............................................................................................................. 10
4. Server Messages .............................................................................................................. 12
5. Client Messages .............................................................................................................. 13
   5.1. Basic Authentication Scheme .................................................................................... 14
   5.2. Digest Authentication Scheme .................................................................................. 16
   5.3. NMEA Request Messages ........................................................................................ 16
6. Source-Table ..................................................................................................................... 17
7. References ........................................................................................................................ 21

## Appendix A  Example Source-Table ............................................................................. 22
## Appendix B  Format Specifications ................................................................................. 23
1. RTCM Formats.................................................................................................................. 24
2. Other Formats ................................................................................................................... 25

## Appendix C  Client Example Source Code .................................................................. 26

## PART II  Ntrip Example Implementation ...................................................................... 30
1. NtripServer ........................................................................................................................ 31
2. NtripServerCMD ................................................................................................................ 33
3. NtripClient (GNSS Internet Radio) .................................................................................... 34
4. NtripCaster ........................................................................................................................ 38
   4.1. Status Information ...................................................................................................... 38
   4.2. Administration ............................................................................................................ 42
   4.3. Hardware .................................................................................................................... 48

Downloads ................................................................................................................................ 50
PART I  Networked Transport of RTCM via Internet Protocol

This documentation is separated in two parts. Part I describes system design and protocol details, aiming to define an HTTP streaming technique for the dissemination of GNNS data. Part II describes an example software implementation. While Part I is meant to define the mandatory communication procedures, Part II reports on an example implementation and thus contributes to software availability.
1. Introduction

Differential GPS (DGPS), or actually any Differential Global Navigation Satellite System (DGNSS) in general, improves the achievable positioning accuracy by correcting the pseudo-distances between receiver and satellites or the receiver position. Serious problems in GNSS positioning are caused by ephemeric, tropospheric, ionospheric, and clock errors. DGNSS is based on the comparison of parameters calculated from observations at a reference station and their well-known correct values. The differences can be used to calculate correction data as defined by the Radio Technical Commission for Maritime Services (RTCM). These RTCM data can be sent to a mobile GNSS rover.

With the stop of the degradation of GPS signals (Selective Availability, SA) on May 1 2000, a stationary GPS receiver without a differential correction signal will show a 10-15m average radius “random walk” pattern. The positioning error of the same receiver can be reduced to approximately ±1.0 m by receiving about 50 Byte/s of DGPS correction data in RTCM format.

The validity of corrections depend on the distance between receiver and rover (±0.5 m accuracy at a distance of 50 km). For providing a service covering a large area, a reference station network has to be introduced, which is able to generate a specific real-time correction for any region. The specific corrections can be transmitted over various communication channels, e.g. via radio transmission (LF, MF, HF, UHF), or a mobile communication network using different communication protocols.

This documentation describes a HTTP-based protocol for disseminating RTCM correction data or other kinds of GNSS streaming data to stationary or mobile receivers over the Internet, allowing simultaneous PC, Laptop, PDA, or receiver connections to a broadcasting host via IP Networks including GSM, GPRS, EDGE, or UMTS. The protocol development follows a feasibility study on real-time streaming of differential GPS corrections as described in [2]). Through the given definitions of this paper, and because a major application will be the dissemination of RTCM corrections, the system as a whole presents a format which should be referred to as the Ntrip-Format (Networked Transport of RTCM via Internet Protocol).

As far as DGNSS and Real Time Kinematic GNSS (RTK-GNSS) is concerned, the use of the popular RTCM-104 standard as the streaming data format is recommended whereby sufficient precision is obtained if given correction data are not older than a few seconds. The RTCM-104 standard is used worldwide. Most (if not all) popular GNSS receivers accept RTCM-104 differential correction messages [3].

The basic data streaming is accomplished by TCP/IP protocol stack. Several attempts 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 [4].
2. System Concept

The Hypertext Transfer Protocol (HTTP) is designed as an application-level protocol for distributed collaborative hypermedia information systems but can also be used for linear streaming media. 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 answer with adequate HTTP respond messages.

Ntrip uses HTTP. It is implemented in three programs called NtripClient, NtripServer and NtripCaster, whereby the NtripCaster is the real HTTP (splitter-)server. NtripClient and NtripServer are acting as HTTP client programs.

Concerning message format and status code, the NtripClient-NtripCaster communication is a full compatible HTTP 1.1 communication [1] whereby Ntrip uses only nonpersistent connections. The NtripServer-NtripCaster communication extends HTTP by a new message format which is “SOURCE” and a new status-code which is “ERROR - Bad Password”.

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

This Ntrip system (see Figure 1) consists of

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

![Fig. 1. Ntrip streaming system](image-url)
NtripServers define a source ID called “mountpoint” for every streamed NtripSource. Several NtripClients can access 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 personal may remote 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 need the possibility 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 a 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 have to specify the parameters characterizing an NtripSource/mountpoint in the source-table.
3. System Elements

A DGNSS reference station in its simplest configuration consists of a GNSS receiver, located at a well-surveyed position. As 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 over its system elements NtripServer, NtripCaster, and NtripClient.

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 (GPS/ GPS+GLONASS), location coordinates and other information.

Every single NtripSource needs a unique mountpoint on an NtripCaster.

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 have to be defined by the administrator of the NtripCaster and handed over to the administrators of the involved NtripServers. An NtripServer in its simplest set-up is a computer program, running on a PC, which 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 transportation 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 these virtual reference station represent a single NtripSource which can be transmitted by an NtripServer.

NtripCaster

The NtripCaster is basically a HTTP server supporting a subset of HTTP request/response messages and adjusted to low bandwidth streaming data (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 might be a part of the NtripCaster program. In this case 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.
**NtripClient**

An NtripClient will be accepted and receive data from an NtripCaster, if the NtripClient sends the right request message (TCP connection to the specified NtripCaster IP and listening Port). Concerning message format and status code, the NtripClient-NtripCaster communication is fully compatible to HTTP 1.1, but Ntrip uses only nonpersistent connections.
4. Server Messages

The NtripServer-NtripCaster communication extends HTTP by the additional message format “SOURCE” and the additional status-code “ERROR - Bad Password”. The password is not protected and therefore based like 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 has to connect to the NtripCaster using the IP and specified listening Port of the NtripCaster. This means, that the NtripCaster has to be “up and running” before any source can connect. Before transmitting GNSS data to the NtripCaster using the TCP/IP connection, the NtripServer has to send an Ntrip server message in order to get access and to specify a mountpoint.

This message is shown in Figure 2.

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

**Fig. 2. Server message structure**

- `<password>` = Encoder password of the Caster
- `<mountpoint>` = Caster mountpoint for the Source
- `<product|comment>` = Information about the source agent

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 has to 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.
5. Client Messages

A client’s request is designed as a 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 has to begin with the string NTRIP.

**Fig. 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/1.0
   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 disc), or request a new source-table before requesting each new 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 (not up-to-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, have to be unique on an NtripCaster. Using a Four-character-ID followed by an integer value (e.g. BUC00 for Bukarest) 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:

\[
\Rightarrow \text{GET /BUCU1 HTTP/1.0}
\]

User-Agent: NTRIP GNSSInternetRadio/1.2.0

The HTTP-server will answer a client request according to the HTTP Protocol [5] with

\[
\Leftarrow \text{HTTP/1.0 401 Unauthorized}
\]

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

\[
\Leftarrow \text{Server: NtripCaster/1.0}
\]

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:

\[
\Rightarrow \text{GET /BUCU1 HTTP/1.0}
\]

User-Agent: NTRIP GNSSInternetRadio/1.2.0

Authorization: Basic aHVnb2JibjpodWdvYmVuMTIz
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>

Different from the server password, the client password is coded like the HTTP Basic Authentication Scheme [5] 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” [8] 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 QWxhZGRpbjpvcGVuc2FhZDpcGVuIHNlc2FtZQ==

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 [5].

5.2. Digest Authentication Scheme

For applications in need for more security regarding user authentication, the Digest Access Authentication for HTTP specified in RFC 2069 [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 2069. [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 GGA strings after the HTTP request for a data stream.

If the <nmea> parameter is set to “1” (see Source-table section), the NtripCaster waits for a NMEA GGA string to prepare the data and start sending.

Following is an example for a request of 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>
```
6. Source-Table

The NtripCaster maintains a source-table containing information on available NtripSources, networks of NtripSources, and NtripCasters to be send to an NtripClient on request. Source-table records are dedicated to either

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.

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>

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) in order 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

<table>
<thead>
<tr>
<th>#</th>
<th>Record Parameter</th>
<th>Meaning</th>
<th>Format</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;type&gt; = STR</td>
<td>The following parameters of this record describe a data stream</td>
<td>3 Characters</td>
<td>STR</td>
</tr>
<tr>
<td>2</td>
<td>&lt;mountpoint&gt;</td>
<td>Caster mountpoint</td>
<td>Characters, 100 max.</td>
<td>LEIJ0 LEIJ1 WTZJ0</td>
</tr>
<tr>
<td>3</td>
<td>&lt;identifier&gt;</td>
<td>Source identifier, e.g. name of city next to source location</td>
<td>Characters, undefined length</td>
<td>Frankfurt</td>
</tr>
<tr>
<td>4</td>
<td>&lt;format&gt;</td>
<td>Data format RTCM, RAW, etc.</td>
<td>Characters, undefined length</td>
<td>RTCM 2.0 RTCM 2.1 RTCM 2.3 CMR CMR+ RTCM ADV RTCM SAPOS RAW RTCA</td>
</tr>
<tr>
<td>5</td>
<td>&lt;format-details&gt;</td>
<td>E.g. RTCM message types or RAW data format etc., update rates in parenthesis in seconds</td>
<td>Characters, undefined length</td>
<td>1(1),2(1),3(30) MBEN(1)</td>
</tr>
<tr>
<td>6</td>
<td>&lt;carrier&gt;</td>
<td>Data stream contains carrier phase information</td>
<td>Integer</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Type</td>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Navigation system(s)</td>
<td>Characters, undefined length</td>
<td>GPS, GPS+GLO, EGNOS</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Network</td>
<td>Characters, undefined length</td>
<td>EUREF, IGS, IGLOS, SAPOS, GREF, Misc</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Three character country code in ISO 3166</td>
<td>3 Characters</td>
<td>GER, ITA, ESP</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Position, latitude, north (approximate position in case of nmea = 1)</td>
<td>Floating point number, two digits after decimal point</td>
<td>40.12, -12.14</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Position, longitude, east (approximate position in case of nmea = 1)</td>
<td>Floating point number, two digits after decimal point</td>
<td>10.12, 357.85</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Necessity for Client to send NMEA message with approximate position to Caster</td>
<td>Integer</td>
<td>0, 1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Stream generated from single reference station or from networked reference stations</td>
<td>Integer</td>
<td>0, 1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Hard- or software generating data stream</td>
<td>Characters, undefined length</td>
<td>JPS Legacy E, GPSNet</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Compression algorithm</td>
<td>Characters, undefined length</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Access protection for this particular data stream</td>
<td>1 Character</td>
<td>N, B, D</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>User fee for receiving this particular data stream</td>
<td>1 Character</td>
<td>N, Y</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Bitrate of data stream, bits per second</td>
<td>Integer</td>
<td>500, 5000</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Source-table Element</td>
<td>Meaning</td>
<td>Format</td>
<td>Examples</td>
</tr>
<tr>
<td>---</td>
<td>----------------------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>&lt;type&gt; = CAS</td>
<td>The following parameters of this record describe a Caster</td>
<td>3 Characters</td>
<td>CAS</td>
</tr>
<tr>
<td>2</td>
<td>&lt;host&gt;</td>
<td>Caster Internet host domain name or IP address</td>
<td>Characters, 128 max.</td>
<td>141.74.243.11 euref-ip.ifag.de</td>
</tr>
<tr>
<td>3</td>
<td>&lt;port&gt;</td>
<td>Port number</td>
<td>Integer</td>
<td>80 2101</td>
</tr>
<tr>
<td>4</td>
<td>&lt;identifier&gt;</td>
<td>Caster identifier, e.g. name of provider</td>
<td>Characters, undefined length</td>
<td>ICD-V1.3 Trimble-iGate</td>
</tr>
<tr>
<td>5</td>
<td>&lt;operator&gt;</td>
<td>Name of institution / agency / company operating the Caster</td>
<td>Characters, undefined length</td>
<td>BKG Geo++</td>
</tr>
<tr>
<td>6</td>
<td>&lt;nmea&gt;</td>
<td>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</td>
<td>Integer</td>
<td>0 1</td>
</tr>
<tr>
<td>7</td>
<td>&lt;country&gt;</td>
<td>Three character country code in ISO 3166</td>
<td>3 Characters</td>
<td>GER ITA ESP</td>
</tr>
<tr>
<td>8</td>
<td>&lt;latitude&gt;</td>
<td>Position, latitude, north</td>
<td>Floating point number, two digits after decimal point</td>
<td>40.12 -12.14</td>
</tr>
<tr>
<td>9</td>
<td>&lt;longitude&gt;</td>
<td>Position, longitude, east</td>
<td>Floating point number, two digits after decimal point</td>
<td>10.12 357.85</td>
</tr>
<tr>
<td>10</td>
<td>&lt;misc&gt;</td>
<td>Miscellaneous information</td>
<td>Characters, undefined length</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 3: Format and contents of source-table records describing a network of data streams

<table>
<thead>
<tr>
<th>#</th>
<th>Source-table Element</th>
<th>Meaning</th>
<th>Format</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;type&gt; = NET</td>
<td>The following parameters of this record describe a network</td>
<td>3 Characters</td>
<td>NET</td>
</tr>
<tr>
<td></td>
<td>Tag</td>
<td>Description</td>
<td>Format</td>
<td>Values</td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>&lt;identifier&gt;</td>
<td>Network identifier, e.g. name of a network of GNSS permanent reference stations</td>
<td>Characters, undefined length</td>
<td>EPN, IGLOS, GREF, SAPOS-THR</td>
</tr>
<tr>
<td>3</td>
<td>&lt;operator&gt;</td>
<td>Name of institution / agency / company operating the network</td>
<td>Characters, undefined length</td>
<td>EUREF, IGS, TRIMBLE, ASCOS, GEO++, SAPOS-THR</td>
</tr>
<tr>
<td>4</td>
<td>&lt;authentication&gt;</td>
<td>Access protection for data streams of the network</td>
<td>1 Character</td>
<td>N, B, D, N,B</td>
</tr>
<tr>
<td>5</td>
<td>&lt;fee&gt;</td>
<td>User fee for receiving data streams from this network</td>
<td>1 Character</td>
<td>N, Y, N,Y</td>
</tr>
<tr>
<td>6</td>
<td>&lt;web-net&gt;</td>
<td>Web-address for network information</td>
<td>Characters, undefined length</td>
<td><a href="http://igs.ifag.de">http://igs.ifag.de</a></td>
</tr>
<tr>
<td>7</td>
<td>&lt;web-str&gt;</td>
<td>Web-address for stream information</td>
<td>Characters, undefined length</td>
<td><a href="http://www.epncb.oma.be">http://www.epncb.oma.be</a> none</td>
</tr>
<tr>
<td>8</td>
<td>&lt;web-reg&gt;</td>
<td>Web address or mail address for registration</td>
<td>Characters, undefined length</td>
<td><a href="mailto:euref-ip@ifag.de">euref-ip@ifag.de</a> <a href="http://igs.ifag.de">http://igs.ifag.de</a></td>
</tr>
<tr>
<td>9</td>
<td>&lt;misc&gt;</td>
<td>Miscellaneous information</td>
<td>Characters, undefined length</td>
<td>none</td>
</tr>
</tbody>
</table>
7. References


<table>
<thead>
<tr>
<th>Source Table</th>
<th>STR; TRAK1; Irvine; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.62; 242.20; 0; 0; Ashtech Z-XII3; none; B; N; 5000; none</th>
</tr>
</thead>
<tbody>
<tr>
<td>STR; OEOC1; Silverado; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.77; 242.26; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>STR; FVPK1; Costa-Mesa; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.66; 242.06; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>STR; CAT21; Avalon; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.31; 242.45; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>STR; MJPK1; Santa-Ana-Mountains; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.71; 242.45; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>STR; SCMS1; San-Clemente; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.44; 242.26; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>STR; WHYT1; Lake-Forest; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.67; 242.36; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
<tr>
<td>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</td>
<td></td>
</tr>
<tr>
<td>STR; SBCC1; Mission-Viejo; RAW; MBEN(1); 2; GPS; SCIGN-OCRT; USA-CA; 33.65; 242.34; 0; 0; Ashtech Z-XII3; none; B; N; 7300; none</td>
<td></td>
</tr>
</tbody>
</table>
Appendix B  Format Specifications

The following is additional information on format specifications as introduced through parameter `<format>` and `<format-details>` in the source-table (see Source-table section). Currently a limited number of raw GNSS data formats and RTCM formats for differential GNSS data are considered.

Each source-table record of type “STR” contains keywords to describe the data format and format-details provided in a specific data stream (see Source-table section). The following sections define keywords and format details corresponding to these keywords.

Introducing provider-dependent other format keywords and format details is allowed.
1. RTCM Formats

Differential GNSS corrections are often available in RTCM format. RTCM stands for Radio Technical Commission for Maritime Services. Working Committees called RTCM Special Committees are chartered to address in depth radiocommunication and radionavigation areas of concern. The RTCM Special Committee 104 on Global Navigation Satellite Navigation Systems Services is responsible for defining RTCM recommended Standards for Differential GNSS. Documentations are available from http://www.rtcm.org. The following is mandatory information to be used within Ntrip that is not part of the RTCM standards.

Table 1 defines RTCM keywords to be used as <format> parameter in source-table records of type STR and the minimum message set corresponding to these keywords. A provider may add more messages. The update rate is not defined.

Table 1: RTCM Minimum Message Set

<table>
<thead>
<tr>
<th>Format / Keyword</th>
<th>Comment</th>
<th>Format Details / Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCM 2.0</td>
<td>DGPS</td>
<td>1, 3</td>
</tr>
<tr>
<td>RTCM 2.1</td>
<td>RTK</td>
<td>3, 18, 19</td>
</tr>
<tr>
<td>RTCM 2.3</td>
<td>RTK</td>
<td>18, 19, 23, 24</td>
</tr>
<tr>
<td>RTCM SAPOS</td>
<td>SAPOS Standard</td>
<td>20, 21, 23, 24, 59</td>
</tr>
</tbody>
</table>
2. Other Formats

Most GNSS equipment vendors have their own proprietary formats. Such “raw” GNSS data may be disseminated using Ntrip. Table 2 shows examples for source-table parameters <format> and <format-details> when streaming proprietary formats.

The “Compact Measurement Record” (CMR) format is a publicly documented format accepted by several GNSS equipment vendors. The CMR format is among those providing some interoperability across vendors.

Table 2: Other data formats

<table>
<thead>
<tr>
<th>Format / Keyword</th>
<th>Comment</th>
<th>Format Details</th>
</tr>
</thead>
</table>
| RAW              | Raw observation data from a GNSS receiver | Examples:  
- RT17 (for Trimble receiver)  
- MBEN (for Ashtech receiver)  
- COMPACT (for Topcon/Javad Receiver) |
| CMR              | CMR Standard             | none                                                |
Appendix C  Client Example Source Code

The following is source code of a Linux/UNIX NtripClient program written under GNU General Public License in C programming language. The program reads data from an NtripCaster and writes on standard output for further redirection of data to a file or COM-port. Please note that this program version does not handle potentially occurring interruptions of communication or network congestion situations.

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

#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: 1.2 $";
static char datestr[] = "$Date: 2003/03/25 13:00:00 $";

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

/* option parsing */
#define NO_LONG_OPTS
#define LONG_OPT(a)
#endif

#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:h:p: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[7] = datestr[14];
    datestr[8] = datestr[15];
    datestr[9] = datestr[16];
    datestr[10] = 0;
    if(!res || help)
    {
        fprintf(stderr, "Version %s (%s) GPL\nUsage: %s -s server -u user ...
" " -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;
}
#endif

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[7] = datestr[14];
    datestr[8] = datestr[15];
    datestr[9] = datestr[16];
    datestr[10] = 0;
    if(!res || help)
    {
        fprintf(stderr, "Version %s (%s) GPL\nUsage: %s -s server -u user ...
" " -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;
}
/* 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] & 0x0F) << 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, sizeof(struct sockaddr));
        if(connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) == -1)
            { perror("connect");
                exit(1); }
        if(!args.data)
            { i = snprintf(buf, MAXDATASIZE,
                "GET / HTTP/1.0\r\nUser-Agent: %s/%s\r\n" |
                    "Accept: */*\r\n"
                // "Accept: */*\r\n"
            }
"Connection: close\r\n", AGENTSTRING, revisionstr);

} else
{
  i=snprintf(buf, MAXDATASIZE-40, /* leave some space for login */
  "GET /%s HTTP/1.0\r\n"
  "User-Agent: %s/%s\r\n"
  /* leave some space for login */
  "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
");
  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;
  }
}

if(args.data)
{
  close(sockfd);
}
return 0;
PART II  Ntrip Example Implementation

The System components described in Part I of this documentation are implemented as an example implementation under several operating systems designed with Microsoft Visual Studio C++ 6.0 including Microsoft Visual Studio 6.0 Service Pack 5 for Win32 and the gcc compiler for Linux (UNIX). A modified Icecast version 1.3.11 is used as the basic element: The NtripCaster. The other programs developed are called NtripServer (Windows 98/00/XP) and the “GNSS Internet Radio” (NtripClient, Windows 98/00/XP).

Note that this example implementation is not capable of handling Ntrip’s NMEA Request Messages as necessary for supporting the VRS concept. It is limited to disseminate data streams with <nmea> parameter set to “0” (see Source-table section in PART I).
1. NtripServer

The NtripServer (Windows 98/00/XP) is designed to be a server for a single NtripSource. Basically the NtripServer grabs a serial GNSS byte stream (COM-port) from a GNSS receiver at a reference station and sends it off over an Internet TCP connection to the NtripCaster. The start window is shown in Figure 4.

Mind that the NtripServer cannot handle a proxyserver. When using it in a proxy-protected Local Area Network (LAN), a TCP-relay has to be established connecting the proxyserver and the NtripCaster. Establishing the Internet connection for an NtripServer by using an Internet Service Provider (ISP) is an alternative.

![NtripServer window](image)

**Fig. 4: NtripServer window**

A server administrator needs to set all relevant parameters before the transmission will be started. The program includes setting windows to configure the serial byte stream (COM-port) from a GNSS receiver and the NtripCaster settings for the data output towards the NtripCaster (see Figure 5). Note that it is recommended to use a high baud rate (e.g. 19200) if supported by hardware and software. Server password and mountpoint are provided by the administrator of the NtripCaster.

![Setting windows](image)

**Fig. 5: Setting windows - press the button “NtripCaster” (right), press the button “COM-port” (left)**

The program will automatically recognize NtripCaster failures (e.g. NtripCaster is down) and can be configured, as shown in Figure 6, to automatically reconnect a the lost NtripCaster.
This example server implementation is currently an experimental software. The BKG disclaims any liability nor responsibility to any person or entity with respect to any loss or damage caused, or alleged to be caused, directly or indirectly by the use and application of the Ntrip technology (see Figure 7).
2. NtripServerCMD

Like the NtripServer, the command line program NtripServerCMD (Windows 98/00/XP) is designed to be a server for a single NtripSource. Basically the NtripServerCMD grabs a GNSS byte stream via TCP/IP from an IP network address and port number and sends it off over an Internet TCP connection to the NtripCaster.

Note that the NtripServerCMD cannot handle a proxyserver. When using it in a proxy-protected Local Area Network (LAN), a TCP-relay has to be established connecting the proxyserver and the NtripCaster. Establishing the Internet connection for the NtripServerCMD program by using an ISP is an alternative.

The user has to call the program with the following arguments:

NtripServerCMD -q <source IP>:<source port>
   -c <caster IP>:<caster port>:<mountpoint>:<password>
   -r <attempts>:<time>
   -p <protocol file>

Explanations:
- q <>:<> - c <>:<>:<> - r <>:<> Obligatory arguments
  :<password> Optional arguments
  <attempts> Number of attempts for connection, '0' for infinite number of attempts
  <time> Interval between connection attempts in seconds
  -p <protocol file> Optional arguments, specifies the name for protocol file,
  no protocol is created without this argument

Use 'Ctrl + c' to quit the program during execution.
The argument '-h' or '/?' will display “help information” on the screen.

Example:
c:\NtripServerCMD -q 194.20.217.4:1024 -c 129.217.182.50:2101:/FFMT0:itsme -r 10:5 -p prot.txt
3. NtripClient (GNSS Internet Radio)

The NtripClient (named "GNSS Internet Radio") is designed to run on a PC/Laptop (Windows 98/00/XP). It feeds a GNSS receiver, which is connected via serial port (COM-port) to the PC/Laptop. The correction data is received from an NtripCaster. The program will handle the HTTP communication and transfer the received GNSS data e.g. to the serial port. After setting up the session parameters and pressing the start button, the software connects to the desired NtripCaster and writes the received GNSS data stream e.g. to the serial port until the session is stopped.

The basic start window is shown in Figure 8. The program keeps a local file of the last session parameters (IP/port, username:password, COM-port settings, source-table etc.). As for first time uses no recorded session parameters are available, they have to be set by the user. The source-table will be automatically received from the NtripCaster when sending the first request message.

![Fig. 8: GNSSInternetRadio window](image)

*first time use (left), a source-table is available (right)*

Broadcaster IP and Port, user-ID and password as well as reconnection settings are entered through the "Broadcaster Settings" windows (see Figure 9). The administrator of the NtripCaster provides the necessary user-ID and the password. It is generally associated to the usage of one specific data stream, but access to networks of data streams or even all data streams available from an NtripCaster can also be allowed through one single password.

![Fig. 9: Broadcaster Settings window](image)

The "Settings" window enables the user to select an output option. The options available are COM-port, TCP/IP, File, or None (see Figure 10). When using a proxyserver, its IP and port number can be introduced as well in this window.

Although the "GNSSInternetRadio" can be used within a proxy-protected Local Area Network (LAN), users may still experience a blocked communication. This can especially happen if a combination of firewall, proxyserver, virus scanner etc. is in front. A work-around can be
either to establish a TCP-relay (proxyserver - NtripCaster, to be arranged by a LAN administrator) or to access the Internet via an ISP.

Details defining the COM-port communication are to be entered through the “COM-port Settings” window (see Figure 11). The usage of a high baud-rate is recommended (e.g. 19200) if supported by hardware and software.

A port number can be defined through the “TCP/IP Settings” window when choosing the TCP/IP output option (see Figure 12). Note that other programs running on the same host as the “GNSS Internet Radio” can access the data stream via the Local IP Address 127.0.0.1.

Received data may be written into a file. The “File Settings” window enables the user to define a file name (see Figure 13).
Details of data streams are available from the “Source Table” windows. You can find information here about the NtripCaster, the data format provided and the network operating the stream generator. Web-addresses or an e-mail address is given for background information or registration purposes.

Note that this example NtripClient implementation is currently an experimental software. The BKG disclaims any liability nor responsibility to any person or entity with respect to any loss or damage caused, or alleged to be caused, directly or indirectly by the use and application of the Ntrip technology (see Figure 14).
Fig. 15: About GNSS Internet Radio & Copyright
4. NtripCaster

The NtripCaster as the central software component receives data streams from NtripServers as generated by NtripSources. The NtripCaster administrator is responsible for handling mountpoints for NtripSources, passwords, NtripClient access-rights, billing, statistics, etc.

The NtripCaster is based on the Icecast software, developed under GNU General Public License. The purpose of Icecast is to duplicate source data for up to thousand or more simultaneously connected clients accessing up to a few hundred different sources. Network connections between source, Icecast, and clients are based on HTTP/TCP/IP. The Icecast server was originally designed to stream MP3-coded data with bit rates of 32 kbit/s up to 128kbit/s. The NtripCaster is a re-designed Icecast for

- DGNSS corrections (about 0.5 kbit/s per stream) or
- RTK corrections (about 5 kbit/s per stream) or
- Raw GNSS receiver data (about 5 kbit/s per stream).

The NtripCaster does not alter the data.

The NtripCaster can be accessed by a remote administrator using the same port (listening port) as NtripServers and NtripClients. This feature is necessary to run the NtripCaster in a professional manner, especially in an international context like EUREF. The NtripCaster provides an administration console restricted to exclusive persons [admin password], which can be accessed by Telnet. In addition, a status inquiry is possible by using a web browser. This enables to control a remote NtripCaster independent from its physical location. An NtripCaster hosted at a professional Internet Service Provider might be administrated by a person located at a completely different place. The administration functionality allows to organize the available GNSS data streams, set aliases, allow or deny clients (e.g. only accept clients from a specific country), kicking out connections to sources, clients or admins, view real-time statistics (all clients, sources) etc. while operating the NtripCaster.

To limit network costs, which are usually calculated based on the transferred data volume, the administrator can restrict the number N of simultaneous connections (e.g. \( N_{\text{max}} = 1000 \)). There is also a built-in bandwidth-measuring tool inside the caster. When the NtripCaster is using more than a certain bandwidth (e.g. 1 MB/s), no further clients will be accepted until the bandwidth falls below the specified limit. The NtripCaster is able to connect to GNSS data streams on a different NtripCaster and can present itself as an NtripServer. This allows e.g. the installation of a European network of NtripCasters. Therefore, Italian GNSS data streams could be made accessible via a German server, or German GNSS data streams can be made accessible via an Italian server.

Very important are the NtripCaster configuration files rtcmcast.conf and source-table.dat, which are used to set up the behavior of the HTTP-server. The administrator is able to set up the number of clients, sources, overall throughput, password for sources, password for admins, access, main behavior, and more. The *.conf and *.dat files can be edited on the local administrator PC (Windows or Unix) and uploaded to the NtripCaster via an FTP connection.

4.1. Status Information

Authorized administrators (name and password) can receive status information via a protected HTTP session (e.g. Internet Explorer). The NtripCaster Web Status Interface is shown in Figure 15.
More than one system administrator may work at the same time on the NtripCaster. Information on the administrators simultaneously logged into the system is available as shown in Figure 16.

As shown in Figure 17, the administrator can check current setting like encoder_password, client_password, admin_password, max_clients, max_clients_per_source, max_sources, maxadmins, throttle (how much bandwidth the server is allowed to use before client and source access is denied), etc.
Statistic information can be of interest for the administrator when operating the NtripCaster. Mean values characterizing the number of users, the number of data streams, the bandwidth in use and other important parameters are available (see Figure 18).

The system administrator has real-time control over client parameters like listen IP, username, chosen source/mountpoint, connection time, bytes written, errors and used NtripClient (Fig. 19). This information is kept in log files for statistical or billing purposes. NtripClient usernames are only logged and displayed for password-protected mountpoints. For open mountpoints the username “null” is displayed/recorded. Unwanted users can be identified in real-time and kicked out by the administrator using the admin console command window.
The system administrator can also view all currently connected NtripSources (Fig. 20).

Red characters when displaying the source-table (see Figure 21) indicate currently unavailable data streams.
4.2. Administration

The NtripCaster can be remote controlled via the admin console. To access the admin console, a password-protected telnet connection to the NtripCaster (e.g. through communications/listening port 80 or ports 80 and 2101) can be used. After connecting, no prompt will be displayed. Then 'ADMIN [admin password]' followed by two new lines (press enter twice) has to be entered. Substitute [admin password] by the administrator password set in ntripcast.conf or enter it as an NtripCaster command line parameter (see following example).

telnet euref-ip.ifag.de 80
ADMIN adm1npa55
Return
Return

The admin console command window is shown in Figure 22.
The admin console defines following basic commands [Icecast 1.3.11, Manual, http://www.icecast.org, 2002]:

**alias**

The alias command is used for two things. It can be used to add an alias for a local mountpoint so that a stream can be accessed from two mountpoints. Or it can be used to add an alias for a remote NtripCaster stream.

Syntax:

```
alias add <mountpoint> <newmountpoint>
alias del <mountpoint>
alias list
```

**allow and deny**

This adds a hostmask to an internal access lists (acl). It specifies that this hostmask should allow or deny access. A type for this acl can be specified, which can be either all, client, source or admin. It only has affect on the specified connection type.

Syntax:

```
allow|deny <client|source|admin|all> add <hostmask>
allow|deny <client|source|admin|all> del <hostmask>
allow|deny <client|source|admin|all> list
```

Examples:

```
allow client add *.se
allow all del *.netcom.com
deny admin add *.se
deny admin del ap.*.com
allow admin list
```

**admins**
The „admins“ command is used to list the admins connected to the server. It will display one admin connection per line, like:

Admin 341 <d66.ryd.student.liu.se> connected for 1 hours, 8 minutes and 14 seconds. 0 commands issued

A host-mask as an argument can be supplied, and only the admins who match the mask, will be shown.

Syntax:
admins [hostmask]

Example:
admins *.ifag.de

dump
This command can be used to dump a connected stream to a file on disk.
If the filename specified is „close“, then the file will be closed.

Syntax:
dump <source id> <filename|close>

Example:
dump 230 /tmp/stream.rpcm

help
Without arguments, the command „help“ prints a list of all commands and what they are used for. With a command as an argument, a slightly longer description of what the command does, is printed.

Syntax:
help [command]

Examples:
help kick
help

kick
This command can be used to get rid of some listeners, admins, or sources. A single connection may be kicked out as well as all connections of a certain type matching a hostmask.

Syntax:
kick <id>
kick <-acs> <hostmask>
kick <hostmask>

Examples:
kick 314
kick -a *.com
kick *.badsource.com

listeners
This command displays a list of all connected listeners. The output can be restricted to those listeners matching a specific hostmask.

Syntax:
listeners [hostmask]

Example:
listeners *.ifag.de

**oper**
Most commands on the admin console are restricted to NtripCaster operators. To obtain operator privileges, the „oper“ command with the NtripCaster operator password as argument can be used.

Syntax:
oper <operator password>

Example:
oper cooloperpass

**pause**
The data flow to a client or from a source can be cut with this command.

Syntax:
pause <client or source id>

Example:
pause 314

**quit**
The „quit“ command is used to leave the NtripCaster admin console. This can only be used, when the console is accessed through the remote interface (e.g. using telnet). An optional argument will be used as a signoff message, e.g. displayed to other connected admins.

Syntax:
quit [message]

Example:
quit Bye Bye

**rehash**
This command can be used after changing parameters in the NtripCaster configuration file (ntripcast.conf or source-table.dat). When used with an argument, it will understand this argument as a filename and parse this as a configuration file.

Syntax:
rehash [filename]

Example:
rehash /tmp/newrtcmast.conf
select

"select" is useful when listeners should be transferred from one stream to another. It can either transfer all clients connected to source A, to source B, or all clients connected to source A matching a pattern, to source B, or just one single client with a specified id, to a new source.

Syntax:

select <-a> <source source id> <target source id>
select <client id> <target source id>
select <source source id> <hostmask> <target source id>

Examples:

select 514 23
select -a 43 23
select 23 *.se 43

sources

"sources" is used just like "listeners" to view all the connected sources. It can be used with an optional argument to limit the list to only the sources matching the specified hostmask. It also accepts a large number of options. By default the output is defined by the variable default_sourceopts.

The options are as follows.

i - Connection id
s - Socket descriptor
t - Time of connect
p - Connection ip
h - Hostname
c - Connection state
y - Source type
c - Number of clients
d - Dumpfile
r - Mountpoint priority
M - Source mountpoint
R - Bytes read
W - Show bytes written
C - Total client connects
T - Time connected

Syntax:

sources [hostmask] [-options]

Example:

sources *.apan.com -aCW

shutdown

This command is used to shutdown the server. An optional argument can be introduced which will be used as a shutdown message broadcasted to all connected admins.

Syntax:

shutdown [message]

Example:

shutdown caster_goes_down
**set**

This command is used without arguments. „set“ will display a list of all NtripCaster variables and their current values. It can be used also to change any of those variables if the name is specified and the new value is entered as arguments. Typing „help settings“ will give a short description of all the settings, and a longer description for the settings is available here.

Syntax:

```bash
set
set <variable name>
set <variable name> <new value>
```

Examples:

```bash
set client_timeout
set max_clients 580
```

**stats**

The „stats“ command will print some information about the current number of connections, averages on client connections, bytes read and written, etc. It looks a lot like the output available from the statistics file. As an argument, „hourly“ or „daily“ can be put. Then the averages and totals for the last hour or day will be printed.

Syntax:

```bash
stats [hourly|daily]
```

Example:

```bash
stats hourly
```

**tail**

The „tail“ command can be issued by an administrator to display messages written to the logfile. It's similar to the unix tail command.

Syntax:

```bash
tail
```

Example:

```bash
tail
```

**tell**

An admin can send a message to all other connected admins using the „tell“ command. Any argument will be send to all admins.

Syntax:

```bash
tell <message>
```

Example:

```bash
tell new NtripSource
```

**unpause**

The „unpause“ command unfreezes a client or stream previously paused with the „pause“ command.
Syntax:
unpause <client or source id>
Example:
unpause 314

**uptime**
The „uptime“ command is similar to the Unix uptime command. It prints the number of days, hours, minutes and seconds the NtripCaster is up and running.
Syntax:
uptime
Example:
uptime

**list**
This command is used to list all connections, sources, admins, or clients.
Syntax:
list [hostmask]
Example:
list *.toomanyarguments.com

**relay**
To mount a remote stream on the NtripCaster, or to mount one of your local streams on a remote server, the „relay“ command should be used.
Syntax:
relay pull [-m <localmount>] <url>
Example:
relay pull ifag.de:2101/BUCU0

**auth**
This command is used to display authorization users, mounts and groups.
Syntax:
auth <groups|users|mounts>
Example:
auth groups
auth users

**4.3. Hardware**
Various Linux distributions and hardware platforms may be used to run the NtripCaster example implementation. The following configuration has been tested in January 2003 and may thus be understood as one possible example.
- Server PowerEdge 1650
- Basic Server Installation Packet, Linux Redhat 7.3
- Intel® Pentium® III Processor 1,40 GHz
- Second Processor, PIII P1650
- 8x DVD-ROM
- Rack Kit for 19” Racks
- Redundant power supply (2x)
- PCI Slots - PE 1650 Riser: 2x 64 Bit/66 MHz
- 1 GB PC133 ECC SDRAM (2x 512 MB)
- 2x18 GB SCSI Hard Disc, 10,000 RPM Ultra3/160
- PERC3/DC PCI RAID Controller 128 MB
- RAID Level 1
- 2x Intel® Gigabit Ethernetadapter Onboard (10/100/1000)
- ERA - Embedded Remote Access onboard integrated
Downloads

The BKG maintains a Web-side in order to keep the interested community up to date about Ntrip developments. This information is available through


The latest version of this document as well as the programs which are part of the example implementation are available through

- NtripDocumentation (latest version of this document)
  http://igs.ifag.de/root_ftp/software/NtripDocumentation.zip

- NtripClient for Windows (GNSS Internet Radio)
  http://igs.ifag.de/root_ftp/software/NtripGNSSInternetRadioWindows.zip

- NtripClient for Windows CE (GNSS Internet Radio)
  http://igs.ifag.de/root_ftp/software/NtripGNSSInternetRadioWindowsCE.zip

- NtripClient for Linux, simple programming example
  http://igs.ifag.de/root_ftp/software/NtripLinuxClient.zip

- NtripServer (reading from serial port)
  http://igs.ifag.de/root_ftp/software/NtripServerWindows.zip

- NtripServerCMD (command-line version of the NtripServer, reading from TCP/IP port)
  http://igs.ifag.de/root_ftp/software/NtripServerWindowsCMD.zip

- NtripCaster
  The NtripCaster developed under GNU General Public License will become available in July 2003.