RTCM STANDARD 10410.1

NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL (Ntrip) - Version 2.0

DEVELOPED BY
RTCM SPECIAL COMMITTEE NO. 104

JUNE 15, 2009

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 10410.1

NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL (Ntrip) - Version 2.0

DEVELOPED BY
RTCM SPECIAL COMMITTEE NO. 104

JUNE 15, 2009

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 blank

Table of Contents

RADIO TECHNICAL COMMISSION FOR MARITIME SERVICES

RTCM STANDARD 10410.1 FOR NETWORKED TRANSPORT OF RTCM via INTERNET PROTOCOL (Ntrip) Version 2.0

1 Scope

1.1 Ntrip

“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) and the Real Time Streaming Protocol (RTSP). Ntrip is a reduced subset of HTTP/RTSP required for real-time data streaming. Ntrip communication usually takes place over HTTP/TCP/IP or RTSP/TCP/IP and RTP/UDP/IP connections. The default port is TCP 2101, but other ports can be used. This does not preclude Ntrip from being implemented on top of any other protocol on the Internet, or on other networks like modem connections or UDP-based transport respectively connection protocols.

Ntrip has been designed for disseminating differential correction data (e.g. in the formats of RTCM Special Committee 104) or other kinds of GNSS streaming data to stationary or mobile users over the Internet.

Ntrip consists of three system software components: clients, servers and casters:

- Caster central server application
- Client user application for data access
- Server provider application for data upload from input source

Ntrip is an open non-proprietary protocol. Major characteristics of Ntrip's Internet based stream dissemination technique are -

based on the popular HTTP/RTSP standards;

easy to implement with limited client or server resources;

not limited to specific data types, but designed as generic streaming protocol;

component separation allows usage of available infrastructure; and

support for mobile and stationary users.

This document describes the protocol at the application level. Knowledge of general TCP/IP connections, socket handling and programming are usually required for implementations. Ntrip is based on certain other open standards. These references are notated in Annex A.

This document describes Ntrip Version 2.0. Major changes compared to Version 1.0 are:

cleared and fixed design problems and HTTP protocol violations;

replaced non standard directives;

chunked transfer encoding;

improvements in header records;

sourcetable filtering; and

RTSP communication.

Detailed remarks on changes between Ntrip Version 1.0 and the current protocol are included at the appropriate places inside the protocol description.

1.2 Purpose

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 satellite clock changes.

RTCM-104 DGNSS standards are used worldwide. Many popular GNSS receivers accept differential correction messages of RTCM-104, 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.

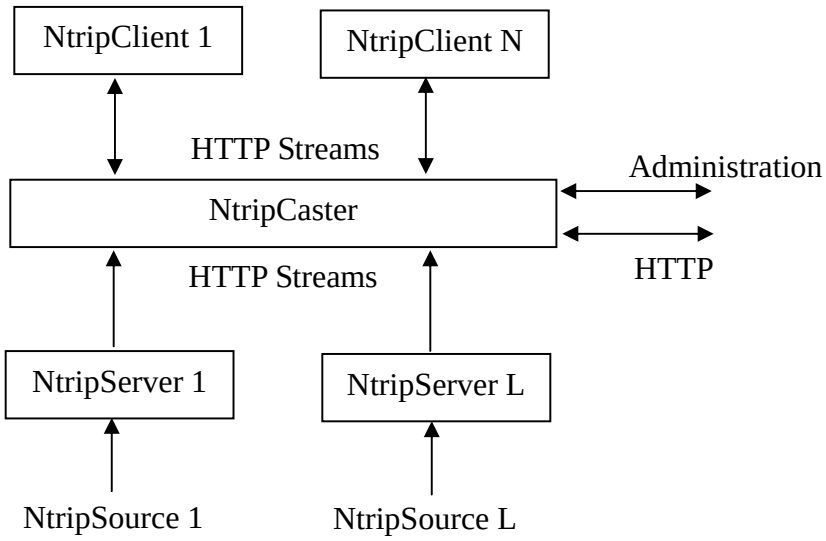
The introduction and deployment of wireless Internet capability has opened up the potential of disseminating these messages over the Internet. This document defines a standard for an HTTP/RTSP-based protocol for the dissemination of DGNSS data (or other kinds of GNSS streaming data) to stationary or mobile receivers over the Internet. It enables simultaneous PC, Laptop, PDA, or receiver connections to a broadcasting host, via mobile IP Networks such as GSM, GPRS, EDGE, or UMTS. Because the primary application is 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).

1.3 System Concept

Ntrip is divided in three different software components: NtripClient, NtripServer and NtripCaster. The NtripCaster is the main server component, whereas NtripClient and NtripServer act as client programs.

This Ntrip system (see figure below) 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 at the NtripCaster.



NtripServers upload the data of an NtripSource to an NtripCaster using a "mountpoint" as identification. Several NtripClients can access the data of a desired data stream in parallel by requesting a source by its mountpoint on the NtripCaster.

If implemented in the NtripCaster program, authorized personnel may remotely control the NtripCaster via password-protected HTTP sessions using an Internet browser or other means of communication. The specific software implementations are not handled in this document. 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 mountpoints.

NtripClients must be able to choose an NtripSource by its mountpoint on the NtripCaster. Therefore a sourcetable (see 6.3) is introduced into, and maintained on, the NtripCaster. Each record of this sourcetable contains parameters describing attributes of a data stream, a network of data streams, or an NtripCaster. Stream attributes are defined at the NtripServer side for each NtripSource.

1.4 System Elements

A DGNSS reference station in its simplest configuration consists of a GNSS receiver located at a well-surveyed position. Because this stationary 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.

1.4.1 Ntrip Server

The NtripServer is used to transfer GNSS data of an NtripSource to the NtripCaster. Every NtripSource needs a unique mountpoint on an NtripCaster. Details of the communication follow in later sections.

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 non-physical reference station. Based on data from a number of reference stations, RTCM corrections are derived for a virtual point at the user's approximate position. Data for this virtual reference station represent a single NtripSource that can be transmitted by an NtripServer.

The NtripSources provide continuous GNSS data (e. g. RTCM-104 corrections) as streaming data. A single source typically represents GNSS data referring to a specific location. Source description parameters as compiled in the sourcetable specify the format in use, location coordinates and other information.

1.4.2 NtripCaster

The NtripCaster is a server supporting a subset of HTTP/RTSP request and 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.

1.4.3 NtripClient

An NtripClient will be accepted by and receive data from an NtripCaster if the NtripClient sends the correct request message. With respect to the message format and status code, the HTTP-based NtripClient to NtripCaster communication is fully compatible to HTTP 1.1, but in this context Ntrip uses only non-persistent connections.

2 HTTP-based communication

The standard TCP communication of Ntrip is based on the HTTP [RFC2616]. Although it is not required to read this documentation to understand Ntrip protocol, it should be consulted in cases of doubt.

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.

The Ntrip protocol consists of a subset of HTTP functions and a few additional informational header lines. The protocol violations of Ntrip 1.0 have been fixed in this version 2.0. Persistent connections are not yet supported in the HTTP part of the Ntrip protocol.

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

Any communication is initiated either by the NtripClient or the NtripServer and responded by the NtripCaster. The basic structure of such a request response communication looks like

```
REQUESTCOMMAND values HTTP/1.1<CR><LF>
Headerline1: values<CR><LF>
Headerline2: values<CR><LF>
<CR><LF>
```

```
HTTP/1.1 CODE Response<CR><LF>
Headerline1: values<CR><LF>
Headerline2: values<CR><LF>
<CR><LF>
```

The sequence <CR><LF> is the sequence of characters 0xD and 0xA and is the standard line termination for HTTP communication. Every request consists of a header line, more or less optional header lines (see 2.6 for header descriptions) and an empty line to end the header section.

After these headers, data may or may not follow (depending on the actual message).

The following examples assume an NtripCaster is running at host ntrip.example.com port 2101 (the default port for Ntrip protocol communication).

2.1 NtripClient to NtripCaster communication

This section describes the majority of communications in the Ntrip system. The NtripClient is any software receiving data streams from the Internet provided by an NtripCaster. This protocol must be implemented in any end user application or device.

The basic tasks of these communications are:

- transfer of system information from NtripCaster to NtripClient (sourcetable)
- request of data by NtripClient (specified with mountpoint, user name, password)
- transmission of requested data from NtripCaster to NtripClient
- handling of error states, wrong requests, etc.

An example NtripClient implementation can be found at [SRCNC].

2.1.1 Requesting sourcetable information

To get an overview of available data sources, a sourcetable (see 6.3) can be requested.

```
GET / HTTP/1.1<CR><LF>
Host: ntrip.example.com<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
User-Agent: NTRIP ExampleClient/2.0<CR><LF>
Connection: close<CR><LF>
<CR><LF>
```

```
HTTP/1.1 200 OK<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Ntrip-Flags: st_filter,st_auth,st_match,st_strict,rtsp<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
Connection: close<CR><LF>
Content-Type: gnss/sourcetable<CR><LF>
Content-Length: 1234<CR><LF>
<CR><LF>
sourcetable data
```

The NtripClient sends the initial request to the NtripCaster. The first line requests the sourcetable and the last line ends the request. The 7 header lines are optional, but recommended minimum.

The NtripCaster sends the requested data as reply. Again all the header lines are optional, but recommended minimum.

For Ntrip1.0 the response of the NtripCaster is a bit different. This response is also used in case of a missing mountpoint instead of an error reply.

```
SOURCETABLE 200 OK<CR><LF>
Server: NTRIP ExampleCaster 2.0/1.0<CR><LF>
```

```
Connection: close<CR><LF>
Content-Type: text/plain<CR><LF>
Content-Length: 1234<CR><LF>
<CR><LF>
sourcetable data
```

2.1.2 Requesting GNSS data

To actually access the wanted GNSS data, the following request must be used.

```
GET /ExampleMountpoint HTTP/1.1<CR><LF>
Host: ntrip.example.com<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
User-Agent: NTRIP ExampleClient/2.0<CR><LF>
Authorization: Basic bnRyaXA6c2VjcmV0<CR><LF>
Connection: close<CR><LF>
<CR><LF>
```

```
HTTP/1.1 200 OK<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
Cache-Control: no-store, no-cache, max-age=0<CR><LF>
Pragma: no-cache<CR><LF>
Connection: close<CR><LF>
Content-Type: gnss/data<CR><LF>
<CR><LF>
data
```

Again, most of the header lines are optional, but recommended. The line “Authorization” is used to transfer user name and password. Section 2.3 explains the authorization message. This line is not required for unprotected data streams. Information regarding the transfer encoding is given in 2.4.

See 2.1.3 for handling of non-physical reference station (user position dependent) data streams.

In violation of HTTP, standard Ntrip Version 1.0 servers will answer with the following line:

```
ICY 200 OK<CR><LF>
```

Any client should be able to detect this answer and assume Ntrip 1.0 transfer (this means no features of Ntrip 2.0 are available). Any caster should answer with this message when the initial request comes from an Ntrip 1.0 only client.

In case of an error (except illegal password) an Ntrip 1.0 caster will send the sourcetable instead of an error reply.

2.1.3 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 non-physical reference station or to determine the best data stream to broadcast. If a streams <nmea> parameter is set to “1” (see 6.3), the NtripCaster waits

for at least one position string to prepare the data and start sending. To support these dynamic data streams, the request for a certain mountpoint may be modified in two ways:

The preferred method of an Ntrip 2.0 NtripClient is to transfer the NMEA position in a header line. To do so, the following additional header line is used:

```
Ntrip-GGA: $GPGGA,040941.00,5034.91174, ... 47.7,M,,*51<CR><LF>
```

The second method also supported in Ntrip 1.0 is to transfer the NMEA position unmodified after the request to the server (e.g. after the empty line).

```
$GPGGA,040941.00,5034.91174, ... 47.7,M,,*51<CR><LF>
```

The NtripClient is allowed to send more than one NMEA GGA string or NMEA strings of other type than GGA at any time.

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.

2.1.4 Requesting filtered sourcetable

To reduce the amount of sourcetable information transferred to the user, an interface to request only parts of the sourcetable is introduced in Ntrip 2.0.

This is an optional feature of Ntrip protocol version 2.0. NtripClients may or may not support it. NtripCasters must support filtered source tables, but may transfer more information than requested by the filter string in case full filter support is not implemented.

The sourcetable request is encoded in the URL. After the normal request for the sourcetable ("/") a question mark and the request string is appended.

```
GET /?requeststring HTTP/1.1<CR><LF>
```

The "requeststring" contains one or more variables together with their values. They are specified in the form variable1=value&variable2=value&variable3=value.

Defined variables are:

- auth The usage of "auth=1" restricts the sourcetable to all elements the user is allowed to access. This also means a correct authorization must be passed together with the sourcetable request.
- strict The usage of "strict=1" forces the NtripCaster to return an error for illegal requests (including a helpful error message). Default is to be as fault-tolerant as possible and return a sourcetable including all requested information (and possibly more) in case of problems.
- match Easy filter method for plain string comparison
- filter Complex filtering method

The two methods "match" and "filter" are based on the same structure. Every field of a sourcetable entry either is a string or an integer value. The request string consists of semicolon separated elements which represent one column of that entry. Empty fields at the end can be omitted. Some examples:

```
GET /?match=CAS;;;;;0;DEU HTTP/1.1<CR><LF>
```

Request all Caster entries in Germany with no NMEA feature.

```
GET /?match=CAS HTTP/1.1<CR><LF>
```

Request all Caster entries.

```
GET /?match=STR;;;;;;;EUREF HTTP/1.1<CR><LF>
```

Request all EUREF streams.

```
GET /?filter=STR;;;;;;;EUREF;;>=50+<=51;>=8.1+<=8.6;;;;;N
HTTP/1.1<CR><LF>
```

Request all EUREF streams between 50 and 51 degree north and 8.1 and 8.6 degree east (see below for encoding issues).

```
GET /?auth=1&match=STR HTTP/1.1<CR><LF>
```

Request all streams the user is authorized to access. This requires additional transmission of authorization information.

The two methods “match” and “filter” are different in their complexity. The “match” method does only a plain string comparison; it only tests if an entry is equal to the given element or not. The “filter” method adds logical operators (multiple operators are evaluated from left to right) and more complexity:

- Logical operators for strings and numbers:
  - Operator NOT to inverse request: !
  - Operator AND to select multiple requirements: +
  - Operator OR to select alternatives: |
- Operators for numbers:
  - Equations: <n (lower than), >n (higher than), <=n (lower or equal), >=n (higher or equal), =n (equal), !=n (not equal)
  - Approximation (nearest value): ~n
- Operators for strings:
  - * means any string (e.g. “*REF” selects EUREF and GREF and “RTCM*” selects any RTCM type).
  - The brackets () can be used to group elements: (EU|G)REF|IGS would select EUREF, GREF or IGS.

Whenever the enhanced features of “filter” are not required, an NtripClient should use the “match” method. It is possible to use multiple filters in one request as described above. The result then is a logical “AND” of all the filters.

To enter the string in the GET line it must be encoded using URL encoding (see chapter 2.4 of [RFC2396]). The request

```
GET /?filter=STR;;;;;;;EUREF;;>=50+<=51;>=8.1+<=8.6;;;;;N
HTTP/1.1<CR><LF>
```

becomes

```
GET /?filter=STR;;;;;;;EUREF;;%3E%3D50%2B%3C%3D51;%3E%3D8.1%2B%3C%3D
8.6;;;;;N HTTP/1.1<CR><LF>
```

after encoding. Encoding more characters than necessary isn't wrong.

Support for sourcetable filtering is optional for NtripCaster implementations. If the filtering is not implemented, the request must be detected nevertheless and handled as a normal sourcetable request. A client can detect if a server supports filtering or not by analyzing the Ntrip-Flags header line.

2.2 NtripServer to NtripCaster communication

The NtripServer has only one purpose. Upload data to the NtripCaster. Thus the protocol is simple. As with the NtripClient, every connection is initiated by the NtripServer and responded by NtripCaster

```
POST /ExampleMountpoint HTTP/1.1<CR><LF>
Host: ntrip.example.com<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Authorization: Basic bnRyaXA6c2VjcmV0<CR><LF>
User-Agent: NTRIP ExampleServer/2.0<CR><LF>
Connection: close<CR><LF>
<CR><LF>
```

```
HTTP/1.1 200 OK<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
Connection: close<CR><LF>
<CR><LF>
```

After the request response pair, the transmission of data from NtripServer to NtripClient follows. Additional optional headers (especially header Ntrip-STR) are described in 2.6. Important notes regarding transfer encoding can be found in 2.4.

The Ntrip 1.0 protocol uses a different request. It uses SOURCE instead of POST and transfers the password (no user name) directly after SOURCE. The header User-Agent is renamed into Source-Agent. The optional header Ntrip-STR is named STR only.

```
SOURCE secret /ExampleMountpoint HTTP/1.1<CR><LF>
Source-Agent: NTRIP ExampleServer/2.0<CR><LF>
<CR><LF>
```

The server either answers with

```
OK<CR><LF>
```

or

```
ICY 200 OK<CR><LF>
```

The Ntrip 1.0 error handling does not follow the standards as described in 2.5. In case of an invalid password the reply is

```
ERROR - Bad Password<CR><LF>
```

or in case of an already used or invalid mountpoint it is

```
ERROR - Mount Point Taken or Invalid<CR><LF>
```

or

```
ERROR - Already Connected<CR><LF>
```

An example NtripSource implementation can be found at [SRCNS].

2.3 Authentication

The authentication for HTTP is described in the document [RFC2617]. For Ntrip 1.0 this is used in NtripClient to NtripCaster communication. Ntrip 2.0 also introduces it in NtripServer to NtripCaster communication. The minimum required for Ntrip is the Basic authentication method.

2.3.1 Basic authentication

A client (either NtripClient or NtripServer) includes the user name and password as a header in the request it sends to the NtripCaster.

```
Authorization: Basic bnRyaXA6c2VjcmV0<CR><LF>
```

The transferred string in this example is “ntrip:secret” in Base64 transfer encoding as described in 6.8 of [RFC2045]), where “ntrip” is the user name and “secret” the password.

In case a mountpoint is protected and no “Authorization” line is transferred or the data is wrong, the server answers with a 401 error code and includes following line in the reply.

```
WWW-Authenticate: Basic realm="/ExampleMountpoint"<CR><LF>
```

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 carefully.

2.3.2 Digest authentication

NtripCasters must support the Basic authentication. The Digest authentication can be supported optionally for applications that require more security concerning user authentication. The sourcetable stream entries contain a flag whether Basic or Digest authentication has to be used. For compatibility, an NtripCaster should also support Basic authentication when Digest is requested except when security consideration prevent the use of Basic authentication.

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 [RFC1321] checksum is compulsory) of the user name, 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 [RFC2617].

2.4 Transfer Encoding

```
Transfer-Encoding: chunked<CR><LF>
```

The chunked transfer encoding is used to transfer a series of chunks, each with its own size information. This allows the transfer of streaming data together with information to verify the completeness of transfer. Every NTRIP 2.0 component must be able to handle this transfer encoding. Chunked transfer encoding is recommended for transfer of stream data (caster to client and server to caster). It is described in chapter 3.6 of [RFC2616]. The basic idea is to send the size of the following data block before the block itself as hexadecimal number.

The following would be an example of a 14 (hexadecimal E) and a 19 (hexadecimal 13) byte string transferred in chunked transfer encoding.

```
E<CR><LF>
TEST TEST TEST<CR><LF>
13;extension<CR><LF>
TEST TEST TEST TEST<CR><LF>
```

The HTTP standard allows extensions in the transfer encoding head line (separated by a semicolon from the byte number), which should be ignored if unknown. The exact definition can be found in HTTP standard. For Ntrip using software, it is only necessary that extensions start after a semicolon and should be ignored. The third line of above example contains such an extension. The word “extension” is a placeholder for the text to be ignored.

An example of a starting RTCM 2 data transmission from an NtripCaster may look like following.

```
HTTP/1.1 200 OK<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
Cache-Control: no-store, no-cache, max-age=0<CR><LF>
Pragma: no-cache<CR><LF>
Connection: close<CR><LF>
Transfer-Encoding: chunked<CR><LF>
Content-Type: gnss/data<CR><LF>
<CR><LF>
2D<CR><LF>
fAFC~cM{x|}@pT\PrgRi@|`VjmMC@RAKFeTl}_Tdoxgju<CR><LF>
1E<CR><LF>
Y~x|K\rLAHhUPzCQAjoY\EDn`pp~Uj<CR><LF>
1E<CR><LF>
fAGCtcir~gWjoEYQAjo|cz{Qzpp~UO<CR><LF>
...
```

2.5 Error handling

Whenever a client (NtripServer or NtripClient) issues a wrong request the NtripCaster responds with an error code.

```
HTTP/1.1 code text<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
Content-Type: text/html<CR><LF>
Connection: close<CR><LF>
<CR><LF>
long text
```

There are 3 fields, which change according to the error condition. Most important is the code field. The text directly after the code is a description field and the long text is an (optional) better description (usually in HTML format). The following table lists a set of errors a client may expect and their meaning. See chapter 6.1.1 and chapter 10 of HTTP documentation [RFC2616] for a more detailed description of the status codes.

Code	Short text	Description
200	OK	everything was fine
401	Unauthorized	No or wrong authorization (see also header WWW-Authenticate)
404	Not Found	Mountpoint of request not found (see 2.1.1 for Ntrip 1.0)
409	Conflict	Mountpoint already in use by another NtripServer
500	Internal Server Error	e. g. some internal errors
501	Not Implemented	e. g. Requested function not implemented in NtripCaster
503	Service Unavailable	e. g. in case of NtripCaster overload or bandwidth limitations

Error handling in Ntrip 1.0 does not always follow this standard and is described at the appropriate places.

2.6 List of header lines

Header lines are used to transfer additional information in a request or response. These can be divided in three major groups.

- Header lines which must be supported by the receiver in a specific protocol version
- Optional or informational header lines
- Lines which can be supported or may be ignored (optional features).

The line types are not used by the three Ntrip components equally. Some are used only by the NtripCaster, some by the NtripServer or NtripClient and some by multiple components. This section contains a detailed discussion of the header lines and their usage. More detailed information can be found in HTTP documentation.

For each header line and Ntrip component, a note is given whether the header line is required, recommended, optional or not used in sending. Note that the receiving end needs to know how to handle these lines (e.g. Transfer-Encoding is used by NtripCaster and NtripServer, thus NtripClient and NtripCaster both need to know how to handle the line when receiving it).

2.6.1 Standard HTTP

`Authorization: Basic bnRyaXA6c2VjcmV0<CR><LF>`

Caster: not used Server: required Client: required

To transfer authorization information to the caster, this header is used. See also 2.3.

`Cache-Control: no-store, no-cache, max-age=0<CR><LF>`

Caster: recommended Server: optional Client: not used

Tell proxy servers to disable caching. See also Pragma header. Informational on receiving side.

`Connection: close<CR><LF>`

Caster: recommended Server: recommended Client: recommended

Tell the receiving component that persistent connections are not used. Informational on receiving side for Ntrip software.

Content-Length: *1234*<CR><LF>

Caster: optional	Server: not used	Client: not used

This header specifies the amount of bytes following after the end of the header sections. This cannot be used for the streaming data as byte count is not known. For sourcetable transfer and error codes it is recommended. Informational on receiving side.

Content-Type: *gnss/data*<CR><LF>

Caster: recommended	Server: not used	Client: not used

The header is used to inform about the data type of transferred data. Types used by NtripCaster may be: “text/html” e.g. for error messages, “text/plain” for texts (e.g. sourcetable) and the two special NTRIP 2.0 types “gnss/sourcetable” and “gnss/data”. Informational on receiving side.

Date: Tue, *01 Jan 2008 14:08:15* GMT<CR><LF>

Caster: recommended	Server: optional	Client: optional

The date format is documented in chapter 3.3 of HTTP documentation [RFC2616]. Informational on receiving side.

Host: *ntrip.example.com*<CR><LF>

Caster: optional	Server: recommended	Client: recommended

This line contains the host name of the request. It is required for virtual hosts (multiple different instances of an NtripCaster at one server with same port). Informational on receiving side, but required for virtual casters.

Server: NTRIP *ExampleCaster/2.0*<CR><LF>

Caster: recommended	Server: not used	Client: not used

The server identification string sent by the NtripCaster. It should consist of the following sequence: “NTRIP” + space + SoftwareName + “/” + Version (+ space + additional text). It is informational on receiving side.

For Ntrip 1.0 this line follows the sequence: “NTRIP” + space + SoftwareName + space + Version + “/1.0”. The version after the slash contains the Ntrip version 1.0.

Pragma: no-cache<CR><LF>

Caster: recommended	Server: optional	Client: not used

Tell proxy servers to disable caching. See also Cache-Control header. Informational on receiving side.

Transfer-Encoding: chunked<CR><LF>

Caster: recommended	Server: recommended	Client: not used

This header specifies the used transfer encoding. See 2.4. The receiving end of the connection must be able to handle the encoding correctly.

User-Agent: NTRIP *ExampleServer/2.0*<CR><LF>

Caster: not used	Server: recommended	Client: recommended

The client identification string send by NtripClient or NtripServer. It should consist of the following sequence: “NTRIP” + space + SoftwareName + “/” + Version (+ space + additional text). For NtripServer protocol version 1.0 the string Source-Agent is used. Informational on receiving side.

WWW-Authenticate: Basic realm="*/ExampleMountpoint*"<CR><LF>

Caster: required	Server: not used	Client: not used

This header is required for type 401 error responses. See also 2.3.

2.6.2 Ntrip specific

Ntrip-GGA: *$GPGGA,040941.00,5034.91174, ... 47.7,M,,*51*<CR><LF>

Caster: not used	Server: not used	Client: recommended

This line is used to transfer an initial user position to the NtripCaster so virtual reference stations can be calculated. It is preferred to use this header line in NTRIP version 2.0.

Ntrip-Version: Ntrip/2.0<CR><LF>

Caster: recommended	Server: recommended	Client: recommended

Inform the receiving end about the Ntrip version used. Currently 1.0 and 2.0 are possible. Required for Ntrip 2.0 to select proper communication protocol.

Ntrip-STR: *;ExamplePlace; ... ;none;N;N;400;none*<CR><LF>

Caster: not used	Server: recommended	Client: not used

This is used to transfer a sourcetable entry together with the uploaded data. The supplied string starts at entry 3 of STR entry of the sourcetable (skipping line identifier and mountpoint). For Ntrip 1.0 the header is STR instead of Ntrip-STR.

Ntrip-Flags: <*flag1*>,<*flag2*>,...<CR><LF>

Caster: recommended	Server: not used	Client: not used

This is used to specify the capabilities of the Ntrip programs. Each element included in this header specifies one optional or extended feature. Currently only the sourcetable filtering is defined this way. Private expansions of implementers are allowed as well. These must follow the form “x_companyname_value” to prevent conflicts with future expansions. Any caster should transfer this header line at least in each sourcetable request.

Currently defined are the following flags:

- st_auth Element “auth” of sourcetable filtering is supported.
- st_match Element “match” of sourcetable filtering is supported.
- st_filter Element “filter” of sourcetable filtering is supported.
- st_strict Element “strict” of sourcetable filtering is supported.
- rtsp RTSP/RTP protocol is supported
- plain_rtp RTP only protocol is supported

2.6.3 Obsolete Ntrip 1.0 headers

The Ntrip 1.0 protocol uses some headers which are no longer recommended to be used in newer implementations (except for downwards compatibility).

STR: ***;ExamplePlace; ... ;none;N;N;400;none*** **<CR><LF>**

Caster: not used Server: optional Client: not used

Ntrip 1.0 uses STR instead of the header Ntrip-STR.

Source-Agent: NTRIP ***ExampleServer/2.0*** **<CR><LF>**

Caster: not used Server: recommended Client: not used

Ntrip 1.0 uses Source-Agent in NtripServer instead of User-Agent.

3 RTSP(RTP) communication

The stateless UDP protocol has some differences to the TCP protocol:

- reduced amount of transmission overhead (reduced data, cheaper costs)
- no transmission control (out of order data, lost data)
- no end of transmission detection (control connection required)

To handle the problems of UDP, a combined method using a TCP control connection and UDP data transfer is used. Adapted variants of the RTSP [RFC2326] and RTP [RFC3550] protocols are used for Ntrip:

- RTSP control connection is necessary for billing purposes and end of transmission detection
- RTSP protocol is compatible with the HTTP instance of TCP connections
- RTP allows detection and fixing of out of order data packages
- RTP allows detection of lost data
- Sourcetable request done with normal HTTP request
- The NtripCaster must support persistent RTSP connections only

Note that the protocol described in this section is optional. A caster has implemented this optional protocol when “rtsp” method is included in “Ntrip-Flags” header.

3.1 NtripClient to NtripCaster communication

3.1.1 Setup

The connection protocol consists essentially of 3 states. This first communication establishes a connection using the RTSP protocol.

SETUP rtsp://ntrip.example.com/*ExMountpoint* RTSP/1.0<CR><LF>
CSeq: 1<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Ntrip-Component: Ntripclient<CR><LF>
Authorization: Basic *bnRyaXA6c2VjcmV0*<CR><LF>
User-Agent: NTRIP *ExampleServer/2.0*<CR><LF>
Transport: RTP/GNSS;unicast;client_port=*4588*<CR><LF>
<CR><LF>

Most of the elements here are equal to the TCP variant. New elements are the Ntrip-Component header specifying the element of the Ntrip software and the Transport header. This header includes a client port. This client port can be randomly chosen and represents the UDP port, where the client wants to receive the UDP data. The Authorization line is only required for protected data streams.

The server answers this request with

```
RTSP/1.0 200 OK<CR><LF>
CSeq: 1<CR><LF>
Session: 47112344<CR><LF>
Transport: RTP/GNSS;unicast;client_port=4588;server_port=6256
<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
<CR><LF>
```

This answer contains 2 new elements. The first one is a session number, which needs to be used in any further communication. Second is the UDP port number the caster will send the data from.

3.1.2 Start transfer

To start the data transfer a second request is used:

```
PLAY rtsp://ntrip.example.com/ExMountpoint RTSP/1.0<CR><LF>
CSeq: 2<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

and this is answered with

```
RTSP/1.0 200 OK<CR><LF>
CSeq: 2<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

The data is now transferred from the caster using the server port to the client using the client port. See 3.6 for information about the RTP data stream.

> NOTE: See 3.7 for handling firewall systems. Usually, a special UDP packet is required before sending the above request to allow connections through firewalls.

3.1.3 End transfer

To stop the data transfer, the following request is used:

```
TEARDOWN rtsp://ntrip.example.com/ExMountpoint RTSP/1.0<CR><LF>
CSeq: 3<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

and this is answered with

```
RTSP/1.0 200 OK<CR><LF>
CSeq: 3<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

After this sequence the data transfer stops.

Closing the control connection also stops data transfer.

3.2 NtripServer to NtripCaster communication

The initial data setup is equal to the NtripClient communication (with the exception that authorization is required always). See 3.1.1 for information about this initial request. Note that the header “Ntrip-Component” must be set to “Ntripserver”.

To start the data transfer to the caster a second request is used:

```
RECORD rtsp://ntrip.example.com/ExMountpoint RTSP/1.0<CR><LF>
CSeq: 2<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

and this is answered with

```
RTSP/1.0 200 OK<CR><LF>
CSeq: 2<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

The data is now transferred from the caster using the server port to the client using the client port. See 3.6 for information about the RTP data stream. The special initial packet for firewall handling in not required for NtripServer communication.

The connection end is equal to NtripClient communication. See 3.1.3 for more information.

3.3 Keep-Alive handling

After data transfer with RTP has been started, the RTSP protocol itself has to transfer no data for a long time. This has two disadvantages:

- Proxy servers may close the connection after a long time of inactivity
- The end of a TCP connection cannot always be detected (e.g. disconnected hardware).

To solve this, a request is used to tell both communication partners that the connection is still alive. This request is always issued by NtripServer or NtripClient and sent to the NtripCaster in regular intervals.

```
GET_PARAMETER rtsp://ntrip.example.com/ExMountp RTSP/1.0<CR><LF>
CSeq: 42<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

This is answered with

```
RTSP/1.0 200 OK<CR><LF>
CSeq: 42<CR><LF>
Session: 47112344<CR><LF>
<CR><LF>
```

This request should be sent every 30 seconds. If an NtripCaster does not receive such a request for more than 1 minute it may close the connection. The NtripServer or NtripClient close the connection when sending the request fails.

The Keep-Alive packet can be used to transfer updated header lines to the NtripCaster. The Ntrip-GGA header must be supported by any NtripCaster.

3.4 Error handling

Whenever a client (NtripServer or NtripClient) issues an incorrect request, the NtripCaster responds with an error code.

```
RTSP/1.0 code text<CR><LF>
Cseq: 2<CR><LF>
Session: 99999<CR><LF>
Ntrip-Version: Ntrip/2.0<CR><LF>
Server: NTRIP ExampleCaster/2.0<CR><LF>
Date: Tue, 01 Jan 2008 14:08:15 GMT<CR><LF>
<CR><LF>
```

There are 4 fields which change according to the error condition. Most important is the code field. The text directly after the code is a description field. The sequence number and the session number are the value used in the client request. Each of these are only given when the initial request includes these lines.

The following table lists a set of errors a client may expect and their meaning. See chapter 6.1.1 and chapter 10 of HTTP documentation [RFC2616] for a more detailed description of the status codes.

Code	Short text	Description
200	OK	everything was fine
401	Unauthorized	No or wrong authorization (no header WWW-Authenticate)
404	Not Found	Mountpoint of request not found
409	Conflict	Mountpoint already in use by another NtripServer
454	Session Not Found	UDP transfer protocol session not found
500	Internal Server Error	e. g. some internal errors
501	Not Implemented	e. g. Requested function not implemented in NtripCaster
503	Service Unavailable	e. g. in case of NtripCaster overload or bandwidth limitations

3.5 List of header lines

This section only explains the special headers required for the RTSP transport. See 2.6 for more information about the other headers.

3.5.1 Standard RTSP

RTSP uses some additional header lines not used in HTTP communication:

```
CSeq: 1<CR><LF>
```

Caster: required Server: required Client: required

Request sequence number. This number is increased by 1 for every request sent to the caster. The caster answers the request with the same number.

```
Session: 47112344<CR><LF>
```

Caster: required Server: required Client: required

After the initial setup, all the request of the RTSP connection need to include the session number.

```
Transport: RTP/GNSS;unicast;client_port=1234;server_port=4321
<CR><LF>
```

Caster: required Server: required Client: required

The header selects the transport protocol. For Ntrip, the only two variables are the two port numbers. See protocol description regarding how they are selected.

The header lines “Authorization”, “Date”, “Server” and “User-Agent” are used in RTSP as well. See 2.6.1 for more information.

3.5.2 Ntrip specific

RTSP uses an Ntrip specific header line not used in HTTP communication

`Ntrip-Component: `*`Ntripclient`*`<CR><LF>`

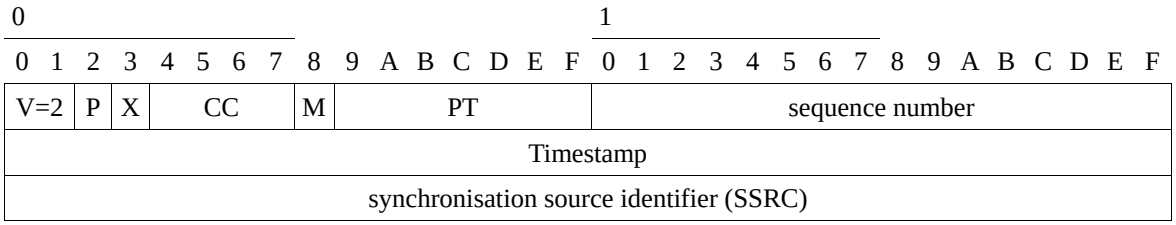
Caster: not used Server: required Client: recommended

Inform the receiving caster about the Ntrip component. The values are either “Ntripclient” or “Ntripserver”.

The header lines “Ntrip-GGA”, “Ntrip-STR” and “Ntrip-Version” are supported for RTSP as well. See 2.6.2 for more information.

3.6 RTP data stream

Each block of the RTP (see [RFC3550]) data stream starts with a header, which consists of 12 bytes for Ntrip and has the following elements:



The fields have the following values:

Version (V)	always 2	identifies RTP version
Padding (P)	0	not used in Ntrip Version 2.0, maybe used in future
Extension (X)	0	not used in Ntrip Version 2.0, maybe used in future
CSRC count (CC)	0	not used in Ntrip Version 2.0, maybe used in future
Marker (M)	0	not used in Ntrip Version 2.0
Payload Type (P)	96	Ntrip media type is 96
Sequence number		value incremented by 1 for each packet, starts with a random value
Timestamp		timer clock rate is 8kHz (resolution 125µs), initial value random
SSRC		identifier for stream (session number of RTSP)

The first 6 elements of this header are constant for any transmission. Future extension may use more RTP features and implementations should be prepared for that. The other 3 elements are initialized with randomly chosen values for the first packet. The sequence number is increased by 1 for each following packet. The timestamp is increased by the elapsed time. The SSRC remains constant. These values are used to test the integrity of the data streams as well as to order the packets (e.g. packet number 12 may arrive after packet number 13). Overflows in these fields have to be handled correctly when checking them.

The packet size of the whole packet including all headers can be up to 1526 byte. It is recommended to choose packets as large as possible to reduce header overhead. On the other side, delaying data to build large blocks destroy the real-time concept of Ntrip. Implementers should select the block size sensible to satisfy both requirements.

3.7 Initial RTP packet for firewall handling

Many computer systems are no longer directly connected to the Internet, but are separated by firewalls, proxy servers and different kinds of intelligent routers. In many cases, connections initiated from the local network to the global Internet are allowed, but connections in the other direction are blocked. The RTSP connection is initiated from the local network, so this is not blocked. But the RTP data connection is started by the server. To convince the firewalls that the connection is really wanted, an initial UDP package is sent from the receiving client UDP port to the UDP port of the caster.

The packet is a normal RTP UDP packet with no data attached. It is equal to the “keep alive” packet of plain RTP protocol. The packet consists thus of 12 bytes (in decimal numbers): “128 96 n n t t t t s s s s” (where “n” is the randomly chosen sequence number, “t” the time stamp and “s” the session number).

After this packet has been sent, the “PLAY” request of RTSP can be sent.

This packet is not required in unprotected systems, but it is always recommended to be sent to support protected systems. It does no harm in unprotected environments. The NtripCaster ignores it.

4 Plain RTP protocol

The plain RTP protocol is a different approach for UDP based communication. This optional protocol is a combination of the HTTP based protocol and the RTSP/RTP

communication. The “Ntrip-Flags” header contains the element “plain_rtp” when it is supported.

To establish a connection from an NtripServer or an NtripClient to an NtripCaster, an RTP packet (see 3.6) must be sent to the UDP port of the NtripCaster. The NtripCaster must listen on this port like it does for HTTP. The SSRC of this block can be chosen freely. The “payload type” is 97 instead of 96. The data part of the block contains a normal HTTP request as described in section 2.

The NtripCaster answers this request with an RTP packet using payload type 97 and the same SSRC. The data component of the packet from the NtripCaster contains a standard HTTP reply. The additional header line “Session” (see 3.5.1) should be included in the reply. If the number specified in this header differs from the used SSRC any subsequent communication must use the new session number as SSRC.

The data part of these payload type 97 packets are normal HTTP communication messages. Except for the additional session header, all information given in the HTTP chapters is relevant for this protocol also. Due to the size limit of RTP packets, the number of used headers may be restricted to a minimum. Ntrip protocol version 1.0 must not be supported when plain RTP is used.

Now the normal RTP data communication starts (sending packets from NtripServer to NtripCaster or NtripCaster to NtripClient). For each of these two communication ends, the following rules apply:

- If no packets have been sent for 20 seconds, a Keep-Alive-Packet must be issued. This is a normal RTP packet with no data attached.
- If no packet has been received for 1 minute, the connection can be closed.
- If the connection should be closed, an RTP packet with payload type 98 and no data attached must be sent. If such a packet is received, the connection is finished.

The UDP protocol has no connection control. Missing packets in the communication must be expected and handled accordingly (especially critical when establishing a connection).

A short list of supported payload types:

payload type	data part	description
96	-	keep alive packet
96	stream	normal data transfer
97	HTTP	HTTP request/response
98	-	connection end

5 Implementation notes

a) In case of connection problems, a client software (NtripServer or NtripClient) must not cause unnecessary network usage due to permanent reconnects. To prevent this, an increasing amount of wait time should be inserted between each retry. Wait amounts in the form of 1s, 2s, 4s, 8s, 16s, 32s (factor 2) or 1s, 2s, 3s, 4s, 7s, 10s, 15s (factor 3/2) have been successfully established. The delay should be limited by an upper value (e.g. 5 minutes or 1 hour).

b) An NtripClient should always use the Ntrip protocol version 2 for requests, but must be prepared to receive answers in level 1 protocol.

c) An NtripCaster must detect requests in protocol version 1 and 2 and react accordingly (using the Ntrip-Version field, a protocol level version 2 can be detected easily). Non-Ntrip request (e.g. by web browsers) should be handled as version 2. To detect this for example, the header User-Agent may be scanned for the case insensitive text “NTRIP”.

d) For NtripServers, the protocol level 2 requires an additional user name. If this user name is specified in the configuration, protocol level 2 may be used, level 1 otherwise.

e) Together with Ntrip 2.0 specification, the implementation of a virtual hosting feature is possible in NtripCasters. Each request sent to a caster should include a “Host” header. A caster may now serve different sourcetables, allow different user groups or different data streams depending on the host name transferred in the “Host” header. Separating sourcetables for different user groups helps to reduce the amount of sourcetable data transferred to the user client. Virtual hosting allows only one NtripCaster instance serving multiple external instances. Previously, this separation was only possible using multiple ports. Note that the “Host” header line is not required. Older Ntrip clients will not transfer it and users of mobile devices may prefer to enter IP addresses instead of host names (numeric keyboards). Reasonable defaults should be possible for these cases.

f) The usage of chunked transfer encoding is recommended for any Ntrip 2 HTTP streaming data transmission from caster to client or server to caster.

g) The order of the header lines in any request or response can be freely chosen. There are some restrictions in HTTP, which are not applicable for the NTRIP subset of used headers. Any software must be prepared to receiver header lines in any possible order. Nevertheless, the request command is always in first line and a header ends with <CR><LF>.

h) Nearly all header lines in this specification are declared to be recommended. Any software should nevertheless include all applicable headers in its own requests or responses and accept minimal headers in received data. Reducing the number of header lines may disable certain (future) functionalities and is not recommended. The communication examples show the minimum usage recommendation and should not be reduced.

6 Sourcetable layout

The NtripCaster maintains a sourcetable containing information on available NtripSources, networks of NtripSources, and NtripCasters, to be sent to an NtripClient on request. Sourcetable records are dedicated to one of the following:

- Data STReams (record type STR),
- CASters (record type CAS), or
- 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 sourcetable 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 sourcetable records is not fixed. Introducing additional data fields is allowed. The last data field always contains "Miscellaneous" information.

The end of the sourcetable is determined by the string: ENDSOURCETABLE

Every line (including the end line) should end with <CR><LF>.

6.1 Caster entry (CAS)

This entry of the sourcetable is used to describe a caster. At least the current caster should be described and a sourcetable and a reference to the rtcm-ntrip.org caster is recommended.

Table 1: Format and Contents of Sourcetable Records Describing a Caster

#	Record Parameter	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
5	<operator>	Name of institution / agency / company operating the Caster	Characters, undefined length	BKG SAPOS
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
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

#	Record Parameter	Meaning	Format	Examples
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

6.2 Network entry (NET)

This entry is used to describe a network. Networks are used for grouping data sources according to their type, data provider or other useful specifications.

Table 2: Format and Contents of Sourcetable Records Describing a Network of Data Streams

#	Record Parameter	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
3	<operator>	Name of institution / agency / company operating the network	Characters, undefined length	EUREF IGS SAPOS
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
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

6.3 Source entry (STR)

This entry describes a data source. For each mountpoint, a record of this type is required.

Table 3: Format and Contents of Sourcetable 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. restricted to “A”-”Z”, “a”-”z”, “0”-”9”, “-”, “.” and “_”	LEIJ0 LEIJ1 WTZJ0
3	<identifier>	Source identifier, e.g. name of city next to source location	Characters, undefined length	Frankfurt
4	<format>	Data format RTCM, RAW, etc. (see 6.3.1 and 6.3.2)	Characters, undefined length	RTCM 2.3 RTCM 3 CMR RAW
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

#	Record Parameter	Meaning	Format	Examples
13	<solution>	Stream generated from single reference station or from networked reference stations 0 = Single base 1 = Network	Integer	0 1
14	<generator>	Hard- or software generating data stream	Characters, undefined length (see [IGSRCVANT])	JPS Legacy E
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

The following provides additional information on format specifications as introduced through parameters <format> and <format-details> in the STR record of the sourcetable. 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 sourcetable 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 other provider-dependent format keywords and format details is allowed.

6.3.1 RTCM Formats

Differential GNSS corrections are often available in RTCM format. Table 4 defines RTCM keywords to be used as <format> parameter in sourcetable 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 4: 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	1, 3, 18, 19, 20, 21,23, 24

	document. If fixed status has not been achieved, messages based on last tentative description as officially documented	
RTCM 3	Messages based on primary document of RTCM Version 3	1004, 1005, 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

6.3.2 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 5 shows examples for sourcetable parameters <format> and <format-details> when streaming such data.

Table 5: 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, meta data, ephemeris, orbits or solutions, see http://binex.unavco.org	none

Annex A
(informative)
Informative References

IGSRCVANT: Y. Bock, W. Gurtner, P. Jamason, G. Mader, C. Meertens, A. Moore, R. Neilan, C. Noll, T. Springer, IGS receiver and antenna table, continuously updated, http://igscb.jpl.nasa.gov/igscb/station/general/rcvr_ant.tab

RFC1321: R. Rivest, The MD5 Message-Digest Algorithm, April 1992, http://tools.ietf.org/html/rfc1321

RFC2045: N. Freed, N. Borenstein, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, November 1996, http://tools.ietf.org/html/rfc2045

RFC2326: H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol (RTSP), April 1998, http://tools.ietf.org/html/rfc2326

RFC2396: T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifiers (URI): Generic Syntax, August 1998, http://tools.ietf.org/html/rfc2396

RFC2616: R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hyptertext Transfer Protocol -- HTTP/1.1, June 1999, http://tools.ietf.org/html/rfc2616

RFC2617: J.Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, HTTP Authentication: Basic and Digest Access Authentication, June 1999, http://tools.ietf.org/html/rfc2617

RFC3550: H. Schulzrinne, S. Casner, R. Frederick, V. Jacobsen, RTP: A Transport Protocol for Real-Time Applications, July 2003, http://tools.ietf.org/html/rfc3550

SRCNC: D. Stöcker, A. Stürze, Open Source Development Repository for POSIX NtripClient, continuously updated, http://sditools.cvs.sourceforge.net/sditools/sditools/ntripclient/

SRCNS: ICD Dortmund, D. Stöcker, A. Stürze, Open Source Development Repository for POSIX NtripServer, continuously updated, http://sditools.cvs.sourceforge.net/sditools/sditools/ntripserver/