Internet-Draft   Ted Stockwell
draft-stockwell-hnews-protocol-00.txt   early morning software
June 17, 1998
Expires six months after above date

HNEWS: An HTTP-Tunneling News Protocol

Status of This Memo

Abstract

This document defines HNEWS (HTTP News Protocol), an HTTP-tunneling news protocol. The NNTP(Network News Transfer Protocol) protocol described in RFC 977 is recast as HTTP CGI (Common Gateway Interface) requests that may be filled by any server that supports CGI. Two new commands are defined for connecting and disconnecting from an HNEWS server. Also, a mechanism for session state management is described. This document assumes that you have a working knowledge of the Common Gateway Interface [2] for running executable content on a web server and some familiarity with the NNTP news protocol [1].
A new URL scheme has also been specified for HNEWS and is described in a separate document [5].

1. Introduction

RFC 977 and RFC 1036 together describe a system for the storage, retrieval, and distribution of news articles. News articles are grouped into logical topics called news groups. It is theoretically possible for to create a news group for any topic of discussion, no matter how specific the topic. However, it is often not practical to

create a specific news group because of the difficulty and expense of gaining access to a news server.
One service that has become easy and relatively inexpensive to obtain is a "web presence provider", an enterprise that will allow a third-party to set up a web site on the provider's web server for a relatively small fee.
The HNEWS protocol defines a way to use a web server as a network news server. It is hoped that this protocol will help popularize news as a means of interaction since it will enable news to be supplied from many more sources.
The basic idea behind this protocol is that a news reader can communicate with a web-hosted news server via HTTP/CGI commands. The news reader sends standard NNTP commands to a CGI script by translating the NNTP command into a CGI request as described later. The CGI script then sends back a standard HTTP response that includes a standard NNTP response to the NNTP command in the original CGI request.
This document includes a section of Java code that unambiguously illustrates the use of the HNEWS protocol.
A new URL scheme has also been specified for HNEWS and is described in a separate document [5].

2. HNEWS commands and responses.

HNEWS commands are CGI requests.
There are three unique attributes of an HNEWS command:
1) The URL of the news server. The news server's URL denotes the name of a CGI script that processes the request and sends back a response.
2) The news.command parameter. The CGI request must include a parameter named "news.command". The news.command parameter contains the NNTP command to execute, formatted exactly as it would be send to an NNTP news server and then "URL-encoded".
3) The news.session.id parameter. Except for the CONNECT command described below the CGI request must include a parameter named "news.session.id". The news.session.id parameter contains an ID, previously returned by the CONNECT command, that identifies the client to the server. Again, the ID must be URL-encoded before being assigned to the news.session.id parameter.
After receiving and executing an NNTP command the CGI script will return an HTTP response. The text of the HTTP response will be the text of the NNTP response to the NTTP command included in the news.command parameter of the original CGI request.
URL-encoding is described in section 2.2 of RFC 1738 [4].

3. Connecting to an HNEWS server.

An NNTP news reader typically connects to an NNTP news server by opening a socket connection to the server given the server's name and the port number on which the server's news service is running. After the socket connection is created the server will send the reader a

response. The connection may be refused by the NNTP server in which case the connection is terminated by the NNTP server after sending a reponse. The response will indicate why the connection was refused.
This process is replaced by sending an HNEWS server a CGI "CONNECT" command.
The CONNECT command has no parameters, the news.command parameter of the associated CGI command is simply set to "CONNECT\n\r".
The first line of the response from the server will be exactly the same as would be returned to an NNTP reader when a session is opened through a port.
If a CONNECT command is unsuccessful(has a status code less than 400) then there will only be one line in the response and the status code at the beginning of the response will indicate the reason for failure.
If a CONNECT command is successful(has a status code greater than or equal to 400) then two additional lines of text are appended to the response. The line of the response contains the news session ID (not including the line feed and carriage return characters at the end of the line). This ID identifies the session to the server and must be passed as a parameter in all future CGI requests (see above). The last line of the response is an NNTP multi-line response terminator, ".\n\r".
The CONNECT command MUST be sent before any other commands.

4. Translating NNTP commands to HNEWS commands.

An NNTP command is translated into an HNEWS CGI request by assigning the text of the command to the news.command parameter of the CGI request.
The news.command parameter should contain the exact text of the original NNTP request. The NNTP command should, of course, be "URL encoded" before assigning it to the news.command parameter.
The HNEWS command may be sent as a PUT command or a GET command.
The HNEWS command must also include the news.session.id parameter.

5. Translating HNEWS news responses to NNTP news responses.

An HTNP CGI script will respond with an HTTP response that includes the exact text of an appropriate NNTP response to the NNTP command in the original HNEWS command.
The type of the HTTP response is set to text/plain (as opposed to text/html).

6. Disconnecting from an HNEWS server.

An HNEWS news reader may disconnect from an HNEWS server by sending a "DISCONNECT" command along with the news.session.id parameter.
The DISCONNECT command has no parameters, the news.command parameter of the associated CGI request is simply set to "DISCONNECT\n\r".
The DISCONNECT commands informs the server that it may release any resources allocated for the session.
If the reader never sends such a command then the server will destroy the session after a certain amount of time, usually 30 minutes.

7. Example conversation

This section contains an example conversation between an HNEWS reader and an HNEWS server coded in Java. The intention of including this code sample is to illustrate the use of HNEWS protocol in a totally unambiguous fashion. In this conversation the HNEWS reader connects to the server, sets the current news group to "the.newsgroup", retrieves the first article from the group, posts a new article to the group, and then disconnects from the server. Code is included for both the client and server side of the conversation.

7.1 Example client code, HNEWS_CLIENT.java

```
import java.net.*;
import java.io.*;

/**
 * This class has a sample conversation with an HNEWS server.
 */
public class HNEWS_CLIENT
{
    /**
     * A fictional HNEWS server's URL.
     */
    URL url= null;

    /**
     * The news session's ID
     */
    String id= null;

    static public void main(String[] args)
    throws MalformedURLException, IOException
    {
        (new HNEWS_CLIENT()).run();
    }

    HNEWS_CLIENT() throws MalformedURLException
    {
        url= new URL("http://127.0.0.1:8080/servlet/HyperNewsServlet");
    }

    /**
     * This method implements a sample conversation with a HNEWS server.
     * This method does not check the status codes of NNTP response, it
     * assumes that all commands are executed successfully
     */
    public void run() throws IOException
    {
        // connect to the server
        String reply= exec("CONNECT\r\n");

        // save the id on the second line of the response
        int i= reply.indexOf("\r\n");
        id= reply.substring(i + 2, reply.indexOf("\r\n", i + 1));

        // set the current group to "the.newsgroup"
        exec("GROUP the.newsgroup\r\n");

        // get the first article
        reply= exec("ARTICLE 1");
        String article= reply.substring(
            reply.indexOf("\r\n") + 2, reply.length() - 3);
System.out.println("ARTICLE 1:");
System.out.println(article);

        // send an article
        exec("POST\r\n");
        exec("From: \"ted stockwell\" <emorning@ix.netcom.com>\r\n" +
             "Newsgroups: the.newsgroup\r\n" +
             "Subject: a sample message\r\n" +
             "\r\n" +
             "this the one and only line in the message\r\n" +
             ".\r\n");

        // disconnect
        exec("DISCONNECT\r\n");
    }

    /**
     * Executes a given NNTP command.
     * Prints the command and the response on the command line
     */
    String exec(String command) throws IOException
    {
System.out.println("SEND:");
System.out.println(command);
System.out.flush();

        // open a connection
        URLConnection connection= url.openConnection();
        connection.setDoOutput(true);

        // send the command
        PrintWriter out= new PrintWriter(connection.getOutputStream());
        try
        {
            /*  if we have previously been given a session id then also
                send the id
            */
            if (id != null)
            {
                out.print("news.session.id=");
                out.print(id);
                out.print('&'); // CGI parameter separator
            }
            out.print("news.command=");
            out.print(URLEncoder.encode(command));
            out.flush();
        }
        finally
        {
            out.close();
        }

        // read the result
        Reader in= new InputStreamReader(connection.getInputStream());
        StringBuffer buffer= new StringBuffer();
        try
        {
            in = new BufferedReader(in);
            for (int c; (c= in.read()) != -1;)
                buffer.append((char)c);
        }
        finally
        {
            in.close();
        }

        String reply= buffer.toString();
System.out.println("RECEIVE:");
System.out.println(reply);
System.out.flush();
        return reply;
    }
}
```

7.1 Example server code, HNEWS_SERVER.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * This class has a sample conversation with an HNEWS client.
 * This class is hard-code to response to the HNEWS_SERVER class
 */
public class HNEWS_SERVER extends HttpServlet
{
    /**
     * same as doGet(req, res);
     */
    public void doPost (HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        doGet(req, res);
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException
    {
        OutputStream streamOut= res.getOutputStream();
        Writer writer= new OutputStreamWriter(streamOut, "8859_1");

        // get the news command
        String[] values= req.getParameterValues("news.command");
        String command= values[values.length - 1];

        /*  If the command is a "CONNECT" command then return a session id.
            We'll always return 1 for the session ID
        */
        if (command.equalsIgnoreCase("CONNECT\r\n"))
        {
            writer.write("200 server ready - posting allowed\r\n");
            writer.write("1\r\n"); // the session ID
            writer.write(".\r\n"); // NNTP terminator
        }
        else if (command.equalsIgnoreCase("GROUP the.newsgroup\r\n"))
        {
            writer.write("211 1 1 1 the.newsgroup group selected\r\n");
        }
        else if (command.equalsIgnoreCase("ARTICLE 1\r\n"))
        {
            writer.write("220 1 <d04321002a@earlymorning.com> ");
            writer.write("article retrieved - head and body follow\r\n");
            writer.write("Message-ID: <d04321002a@earlymorning.com>\r\n");
            writer.write("Date: Wed, 06, May 1998 01:30:33 Central Daylight
Time");
            writer.write("From: \"ted stockwell\"
<emorning@ix.netcom.com>\r\n");
            writer.write("Newsgroups: the.newsgroup\r\n");
            writer.write("Subject: message uno\r\n");
            writer.write("\r\n");
            writer.write("this the one and only line in the message\r\n");
            writer.write(".\r\n");
        }
        else if (command.equalsIgnoreCase("POST\r\n"))
        {
            writer.write("340 send article to be posted.\r\n");
        }
        else if (command.equalsIgnoreCase("DISCONNECT\r\n"))
        {
            writer.write("205 closing connection - goodbye!\r\n");
        }
        /* must be sending the article */
        else
        {
            writer.write("240 article posted ok\r\n");
        }

        writer.flush();
    }

    public String getServletInfo()
    {
            return "Sample server to demonstrate the HNEWS protocol";
    }
}
```

8. References

[1] Kantor, B and P. Lapsley, "Network News Transfer Protocol",
RFC-977, U.C. San Diego and U.C. Berkeley.

[2] D.R.T. Robinson, "The WWW Common Gateway Interface Version 1.1",
Internet-Draft, <draft-robinson-www-interface-01.txt>,
University of Cambridge, 15 February 1996

[3] Horton, M. and R. Adams, "Standard For Interchange of USENET
Messages", RFC 1036, AT&T Bell Laboratories, Center for Seismic
Studies, December 1987.

[4] Berners-Lee, T., Masinter, L. and McCahill, M., Editors,
"Uniform Resource Locators (URL)", RFC 1738, CERN, Xerox Corporation,
University of Minnesota, December 1994.

[5] Ted Stockwell, "The hnews URL scheme",
Internet-Draft, <draft-stockwell-hnews-url-00.txt>,
early morning software, July 1998

9. Author contact information:

Ted Stockwell
early morning software
2222 Preston Lakes Court
Plainfield, IL 60544
emorning@ix.netcom.com