NOTE-DRP

The HTTP Distribution and Replication Protocol

Submitted to W3C August 25, 1997

The latest version of this document:
http://www.w3.org/TR/NOTE-drp
This version of this document:
http://www.w3.org/TR/NOTE-drp-19970825
Editors:
Arthur van Hoff, Marimba Inc.
John Giannandrea, Netscape Inc.
Mark Hapner, Sun Microsystems Inc.
Steve Carter, Novell Inc.
Milo Medin, At Home Corp.

Status of this Document

This document is a NOTE made available by the W3 Consortium for discussion only. This indicates no endorsement of its content, nor that the Consortium has, is, or will be allocating any resources to the issues addressed by the NOTE.

This document is a submission to W3C from Marimba, Netscape, Sun Microsystems, Novell, and @Home. Please see Acknowledged Submissions to W3C regarding its disposition.

Abstract

This document provides a specification of a protocol for the efficient replication of data over HTTP.

Table of Contents

1. Introduction

This document provides a complete description of the HTTP Distribution and Replication Protocol (DRP). The goal of the DRP protocol is to significantly improve the efficiency and reliability of data distribution over HTTP. Here is a more detailed list of goals:

- Provide a widely applicable mechanism for the efficient replication of data and content.
- Improve the efficiency and reliability of caching servers and proxies.
- Significantly improve the efficiency of subscription-based data and content distribution.
- Improved reliability and versioning for the distribution of mission-critical applications and data.
- Create an interoperability standard for replicating content between clients and servers from different vendors.
- Keep the protocol simple to avoid incompatibilities.
- Provide functionality that is complementary to existing standards.
- Provide functionality that is backward compatible with existing HTTP servers and proxies.
- Provide functionality that can be deployed anywhere where HTTP is available today.

The HTTP Distribution and Replication Protocol was designed to efficiently replicate a hierarchical set of files to a large number of clients. No assumption is made about the content or type of the files; they are simply files in some hierarchical organization.

After the initial download a client can keep the data up-to-date using the DRP protocol. Using DRP the client can download only the data that has changed since the last time it checked. Downloading only the differences is much more efficient because data typically evolves slowly over time, and because changes are usually restricted to only a subset of the data. One of the goals of the DRP protocol is to avoid downloading the same data more than once.

The data that is distributed can consist of more than just HTML pages; it can consist of any kind of code or content. The DRP protocol provides strong guarantees about data versioning. This is a requirement for distributing mission critical applications, because getting the correct version of each component is a necessary to ensure a complete application will work correctly. Correct file versioning using existing HTTP requests is problematic because there is currently no reliable mechanism for identifying a particular version of a file.

The DRP protocol uses content identifiers to automatically share resources that are requested more than once. This eliminates redundant transfers of commonly used resources. The content identifiers used in the DRP protocol are based on widely accepted checksum technology.

The DRP protocol uses a data structure called an *index*, which is currently specified using the eXtensible Markup Language (XML). Because the index describes meta data, we anticipate using the Resource Description Framework (RDF), in a future version of the DRP protocol specification. XML is used in the interim because the RDF standard was not finalized at the time of writing.

The DRP protocol relies on existing HTTP/1.1 functionality to achieve better performance when

replicating files, and it is backward compatible with existing HTTP/1.0 and HTTP/1.1 proxies so that it can be deployed immediately. To further improve the download efficiency and caching behavior of HTTP, the proposal introduces new HTTP header information that may become part of the core HTTP protocol specification in the future. Because the DRP protocol is layered on HTTP, it will benefit from ongoing efforts in the HTTP-NG working group.

The DRP protocol is based on technology which was originally developed, implemented, and deployed by Marimba Inc. Although it was originally designed for software distribution, it is widely applicable in many different areas, and it enables the large scale automatic distribution of software, data, and content to many different clients.

2. The HTTP Distribution and Replication Protocol

2.1 Content Identifiers

The DRP protocol uses content identifiers to identify individual pieces of content. A content identifier is a token that can be used to uniquely identify any piece of data or content. It can also be used to determine whether two pieces of content are identical with great accuracy.

A content identifier consists of one or more Uniform Resource Identifiers (URI) separated by commas. The URIs are combined together into a single content identifier and form a globally unique identifier. Here is the syntax of a content-identifier:

```
content-identifier = URI ( "," URI )*
```

Typically a checksum algorithm is used to generate a content identifier for a piece of content. One of the checksum algorithms which can be used is the Message Digest algorithm from RSA. The MD5 algorithm is a well-known algorithm for computing a 128-bit checksum for any file or object. See http://www.rsa.com/pub/rfc1321.txt for details on the implementation of the MD5 algorithm.

The likelihood of two different files producing the same MD5 checksum is very small (about 1 in 2^64), and as such, the MD5 checksum of a file can be used to construct a reliable content identifier that is very likely to uniquely identify the file. The reverse is also true. If two files have the same MD5 checksum, it is very likely that the files are identical.

It is possible to define a Universal Resource Name (URN), which is a kind of URI, for a 128-bit MD5 checksum:

```
MD5-URN = "urn:md5:" base64-number
```

Another checksum algorithm used in the DRP protocol is the SHA algorithm from the National Institute of Standards and Technology. It is similar to the MD5 algorithm but its checksums are 160 bits long, and have different properties. See http://csrc.nist.gov/fips/fip180-1.txt for details on the implementation of the SHA algorithm.

Here is how a URN is created using a 160-bit SHA checksum:

```
SHA-URN = "urn:sha:" base64-number
```

Both MD5 and SHA based URIs use base64 encoding to encode the checksum of the content. The base64 algorithm encodes each 3 bytes of the checksum in 4 characters containing 6 bits of information each. The details of the base64 encoding algorithm are part of the MIME specification.

Note that a base64 number can include '/' and '+' characters. Although these characters are treated specially in the URN syntax specification, it is not necessary to escape them when used in a checksum URN.

If a content identifier contains a URI that refers to a well known checksum algorithm such as MD5 or SHA, it is possible to verify the integrity of the content. If none of the URIs in the content identifier refer to a known checksum algorithm, the content identifier should be treated as an opaque string that can be used for addressing purposes only.

Here are some examples of valid content identifiers:

```
urn:md5:FNG4c6MJLdDEY1rcoGb4pQ==

urn:md5:HUXZLQLMuI/KZ5KDcJPcOA==

urn:sha:thvDyvhfIqlvFe+A9MYgxAfm1q5=

http://www.acme.com/images/foo.gif,urn:md5:FNG4c6MJLdDEY1rcoGb4pQ==

http://www.acme.com/Example/,urn:x-version:5

ftp://www.acme.com/Hello%20World
```

In applications where the possibility of duplicates for a given URN is unacceptable, content identifiers can be generated with the required uniqueness by using multiple appropriate URIs. If for example version numbers are used, it is important to further qualify the content identifier in order to make it globally unique. This can be achieved by including the URL of the object to which the version number applies in the content identifier.

A content identifier is often embedded in an HTTP header, and therefore the URIs in the content identifier are not allowed to contain reserved characters such as spaces or comma characters. All reserved characters should be encoded as specified in the URI specification.

2.2 Index Format

To describe the exact state of a set of data files, the DRP protocol uses a data-structure called an *index*. An index not only describes the hierarchical structure of the files, but it also describes the version, size, and type of each file. An index is a snapshot of the state of a set of files at a particular moment in time.

An index is typically stored in memory as a tree data structure, but in order for clients and servers to communicate this information over HTTP, an index can be described using XML. The index DTD used by the DRP protocol is specified in Appendix A. Using this DTD, here is an example of an index that describes a hierarchical set of files:

```
<?XML VERSION="1.0" RMD="NONE"?>
<index>
<file path="home.html"    size="12345"  id="urn:md5:PEFjWBDv/sd9alS9BYuX0w==">
<file path="layer1.js"    size="32112"  id="urn:md5:W25YCu3toJt3ZsDsHIZmpg==">
<dir  path="images">
<file path="acme.gif"     size="4532"   id="urn:md5:+hbZN5XfU6QAJB1RFl/KSQ==">
<file path="banner.gif"   size="10452"  id="urn:md5:tr3X+oN3r9kqvsiyDSSjjg==">
</dir>
<dir  path="java/classes">
<file path="Scroll.java"  size="14323"  id="urn:md5:xjBkgWouS6p6FTUMIkx/Zg==">
<file path="gui.jar"      size="540321" id="urn:md5:tcUzw0DKut3SiTpmpAsi8g==">
</dir>
</index>
```

2.3 Index Retrieval

A DRP index is retrieved given a URL to the index. The index can be stored in any file and can be retrieved using a normal HTTP GET request. The mime type of the index file allows clients to treat the index as a special file which provides meta information about other files. The client can use the index to automatically download the files that are specified.

The index file can be an ordinary file on an HTTP server, but it can also be generated dynamically. Note that the index isn't necessarily generated from a file system, it could also represent hierarchical data from a different source such as a database.

Once the initial download is complete, a client can update the content by downloading a new version of the index, and comparing it against the previous version of the index. Because each file entry in the index has a content identifier, the client can determine which files have changed and so determine the minimal set of files that need to be downloaded in order to bring the client up-to-date.

Index Base URL

The index file is typically located in the root directory of the file hierarchy it describes. In that case the base URL of the files in the index is the same as the base URL of the index itself. The index file should not be listed in the index itself.

An index can also explicitly specify a different absolute or relative base URL using the **base** attribute of the **index** tag. This allows an index to describe files that are located in a different directory, or even on a different server.

For example, if the index is loaded from the following URL:

```
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.