Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2012

D. Kutscher
M. Stiemerling
J. Seedorf
NEC

October 24, 2011

Design Considerations for a DECADE SDT
draft-kutscher-decade-protocol-00

Abstract

This memo provides some considerations for the design of a specific DECADE protocol.

Table of Contents

1. Introduction

The DECADE architecture specification [refs.decadearch] describes fundamental principles for DECADE (naming, transport, authorization) and identifies a set of core components and conceptual protocols for accessing in-network storage.

A few candidate technologies have been proposed for a concrete protocol specification, such as HTTP-based protocols [RFC2616], WEBDAV [RFC3744], and CDMI [refs.CDMI] for the actual transport/ application layer functionality, as well as the NI URI scheme [refs.ni-core] for an URI format, and OAuth [RFC5849] for an authentication mechanism.

This memo is intended to aid the discussion about how to design DECADE protocols, and how to leverage existing solutions best. It further gives recommendation for a future Standard Data Transport (SDT). These recommendations are labelled with REC_XY, where XY is a sequential number.

[[Text in double square brackets (like this) is commentary.]]

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

2. Conceptual DECADE Protocols

As described in draft-ietf-decade-arch [refs.decadearch], DECADE conceptually consists of two functional building blocks: DRP (DECADE Resource Control Protocol) and SDT (Standard Data Transport).

DRP would provide conveying authorization-relevant information to servers for access control functions. As such, it is not intended as a stand-alone protocol but rather as a scheme that would be used by SDT instantiations, e.g., for passing authorization tokens from an Application Endpoint to a DECADE server. For a concrete specification, a scheme is needed that supports the representation of authorization information. That scheme should be compatible to the SDT instantiations that are specified (and envisioned to be specified). The assumption is that there would be exactly one DRP.

SDT is an actual protocol that Application Endpoints use for communicating with a DECADE server. Furthermore, SDT can also be used for server-to-server communication, i.e., when DECADE servers want to distribute content to other DECADE servers. A DECADE SDT would use an existing transport protocol and possibly an existing application layer protocol such as HTTP or NFS. In fact, the conceptual DECADE SDT interactions that are defined in draft-decade-arch would most likely be mapped to messages, services etc. of such existing protocols, e.g., the SDT GET request would map to a HTTP GET request. The assumption is that there can be different DECADE SDT specifications, i.e., leveraging different underlying protocols. However it is also the assumption that there would be one mandatory SDT.

REC_01: The selected DRP scheme should be compatible to the different envisioned SDT instantiations.

REC_02: There should be one mandatory SDT implementation.

Figure 1: Recommendations

3. Object Naming and Addressing

3.1. Object Naming

draft-ietf-decade-arch [refs.decadearch] outlines requirements and concepts for naming DECADE resources. In essence, a DECADE name should be globally unique (with a high probablity), it should be application independent, and it should provide a name-content binding through the use of content hashes as part of the name. The requirement for using DECADE names which are globally unique with a high probability stems from the envisioned usage of hashes. Hashes typically ensure two items will have different hashes with a certain probability, but there is typically a very limited risk that those 2 items will have the same hash value.

A concrete control specification needs to define the concrete name format and possibly also baseline hashing algorithms. The name format MUST be suitable for use in different possible SDT instantiations.

[refs.ni-core] specifies a URI-based name format for naming objects, e.g., through content hashes. NI URIs fundamentally provide an hash algorithm identifier, the actual digest value and can provide additional information such as object type information or locator hints.

```
ni:///sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc
```

Figure 2: Example: DECADE NI URI

The NI URI in the example above specifies SHA-256 as the hash algorithm, and provides the digest of an object. DECADE implementations can generate such names independently, without requiring any infrastructure (when creating objects), and they can verify the name-content binding by calculating the hash of an received object and by comparing the result to the name that was used for the object.

NI URIs can optionally provide authority information, i.e., information about an authority that may assist applications in accessing the object. DECADE should not require authority information to be present.

The NI format allows the optional specification of media types (of the referenced objects) through the addition of parameters:

```
ni:///sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc?ct=image/jpeg
```

Figure 3: Example: DECADE NI URI with content type

Such information may be present in URIs, but DECADE should not require such information. It is also important to note that parameters are not considered when testing NI URIs for identity.

NI URIs can be mapped to HTTP URIs, and some HTTP URIs can be mapped to NI URIs. This can be useful for deriving a locator for obtaining NI-named objects without explicit specification. The following example depicts an NI URI with an authority part that is mapped to an HTTP URI (using the well-known convention specified in RFC 5785 [RFC5785]).

```
   ni://decade127.example.com/
sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc

   http://decade127.example.com/.well-known/ni/sha-256/
B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc
```

Figure 4: Example: DECADE NI URI mapping to HTTP URI

There are other possibilities to derive the host name part of the HTTP URI (when no autority information is present in the NI URI), e.g., from the application context that the NI URI was used in. For DECADE, we recommend that Application Endpoints that want to refer other Applications to a DECADE object on a specific server (assuming an HTTP-based SDT), provide the server host name as an authority element of the NI URI, as depicted above. It should be noted that this only works with HTTP (or HTTPS)-based SDTs. It is possible to specify additional/alternative locators using the NI parameter mechanisms (which will be described in a future version of this document).

REC_03: There should be exactly one DECADE name format.

REC_04: The DECADE name format must be suitable for use in different possible SDT instantiations.

REC_05: The DECADE names should used the NI URI format.

REC_06: DECADE should allow for different hash algorithms to be used. SHA-256 should be defined as MANDATORY, i.e., all applications that need to validate name-content binding of objects should be able to deal with SHA-256 hash digests.

REC_07: DECADE should allow for different hash algorithms to be used. SHA-256 should be defined as MANDATORY, i.e., all applications that need to validate name-content binding of objects should be able to deal with SHA-256 hash digests.

REC_08: DECADE should allow for different hash algorithms to be used. SHA-256 should be defined as MANDATORY, i.e., all applications that need to validate name-content binding of objects should be able to deal with SHA-256 hash digests.

REC_09: In an application context where SDT==HTTP (or HTTPS), DECADE Application Endpoints should use the authority element in NI URIs to specify a HTTP server name when referring other Application Endpoints to a specific URL.

Figure 5: Recommendations

3.2. Object Addressing

Section Section 3.1 describes how complete objects are potentially named within DECADE. However, it might be also necessary to address parts of a DECADE object, if such objects are accessible in parts. A typical example is the usage of chunks, i.e., equal parts of a file used by peer-to-peer filesharing to exchange data.

This addressing might be required if:

- an object is not completly loaded on a DECADE server, but DECADE clients should be able to retrieve it while the object is being retrieved by the server;

- DECADE clients do only need to access parts of the object, as they have already retrieved some other parts of the object;

o DECADE clients retrieve a particular object from multipe sources at the same time and thus do only require a subset from a particular DECADE server.

REC_10: The DECADE SDT should allow to retrieve parts of an object.
REC_11: The DECADE SDT should allow DECADE servers to specify which parts of an object are available.
REC_12: The DECADE SDT should allow DECADE clients to request single parts or a range of parts from the DECADE server.

Figure 6: Recommendations with respect to Addressing

4. Authentication and Access Control

The DECADE architecture has a concept of token-based authentication and access control. The idea is that Application Endpoints that are referring other Application Endpoints to a DECADE server provide tokens to these other Application Endpoints. Those would use these tokens when communicating with a server, and the tokens would be meaningful to the server for making acess control decisions.

OAuth is one particular candidate mechanism to be used for token-based authentication and access control. (A detailed analysis will be provided in a future version of this document.) A mechanism such as OAuth would be used by HTTP in specific ways, i.e., by using HTTP header fields -- this would be the DRP instantiation for a specific SDT.

Communincating authentication information between Application Endpoints is out of scope for DECADE specifications; it is assumed that this would rely on application-specific protocols. However, there are principally two options:

o authentication tokens in the specific application protocol; or

o authentication tokens in the object identifier that Application Endpoints use to refer other Application Endpoints to a DECADE server.

Including the authentication tokens in the object would provide an application-protocol-independent way for communicating this information between Application Endpoints. The parameter mechanism of NI URIs could be used for that:

```
   ni://decade127.example.com/
sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc?auth=AHFK34F
```

Figure 7: Example: Authentication tokens in NI URIs

For each SDT specification, there needs to be an algorithm to map the authentication token to specific header fields, messages, etc. of the particular protocol.

REC_13: DECADE should use an NI URI parameter for representing authentication information in object identifiers.

Figure 8: Recommendations

5. General SDT Considerations

5.1. Dealing with Application Contexts, Resource Collection and Other Structure

Fundamentally, DECADE is intended to provide access to resources which are distributed in in-network storage servers for different applications. Such resources should be named uniquely across different servers, and the same resource should be accessible at different servers using the same name.

Different servers, different file transfer, and different remote file system protocols may provide different capabilities for organizing resources in hierarchical structures (collections, file system directories etc.). Since DECADE already provides a way to name resources uniquely across different servers and protocols (through the DECADE naming scheme), SDT (and DECADE in general) should not require or rely on any hierarchical name space structure.

Application-specific structure (e.g., collecting all chunks of a specific media resource) should be dealt with on the application layer, i.e., through the use of "torrent files" or index files that reference the DECADE resources.

Similarly, DECADE resources from different application contexts should not be distinguished by additional name components, direcory names etc., since the DECADE naming scheme already provides for a unique naming of resource across application contexts.

Consequently, any operations on remote file system structures, collections etc. should be orthogonal to DECADE and not be supported by SDT. Specific protocols that an SDT instantiation leverages may provide support for that, but we recommend that such operations are considered out of scope for SDT.

5.2. Server-to-Server Communication

The DECADE architecture [refs.decadearch] describes the operation of Server-to-Server Protocols, which are intended to enable DECADE servers to distribute data objects to other servers, without the need of Application Endpoint interaction. One possible way of operation is that an Application Endpoint (client) would upload a data object to a DECADE server, and that server would then upload the object to one or more other servers, thus acting as a client to those other servers. In addition, an Application Endpoint would also be able to request a DECADE server to download the object from another specified server itself.

For specifying a concrete SDT, some design questions need to be taken:

- o Is it possible to specify only one or multiple target DECADE servers?
- o Most HTTP-based protocols do not support requesting/configuring server-to-server communication natively. We recommend this feature be implemented without changing/extending those protocols.

5.3. Recommendations

REC_14: DECADE should not assume any structure (collections, containers, directories) on DECADE servers.

REC_15: DECADE object identifiers should be flat labels.

REC_16: It should be possible for DECADE server to distribute objects between servers using SDT. An SDT instantiation should provide a corresponding mechanism.

REC_17: DECADE should define a way to specify (control) the distribution of objects between servers.

REC_18: Server-to-Server communication should not require the introduction of new HTTP request types (for HTTP-based SDT).

Figure 9: Recommendations

6. CDMI Considerations

CDMI [refs.CDMI] has been considered as a candidate basis for an DECADE SDT instantiation. This section discusses a few aspects and potential issues for adopting CDMI.

In general, the assumption is that CDMI (as a certain way to leverage HTTP for accessing and managing cloud data) can be used for DECADE. Since CDMI has many features (namely the data management features) that are not required by DECADE, we assume that a CDMI-based SDT specification would specify a subset of CDMI and specify a list of requirements for implementations on how to use the mechanisms of the subset in detail.

6.1. CDMI Content Type Operations

CDMI provides uploading/downloading/deleting etc. data with CDMI content types and with non-CDMI content types. CDMI content type operations use JSON to encode objects (and meta information), i.e., PUT requests would encode the data object in JSON, and response messages to GET requests would also encode the returned object in JSON. Non-CDMI content type operations may also use JSON for encoding certain information, for example for data object meta information, but the object itself would be transmitted directly in message bodies (as non-CMDI web servers would do).

A CDMI-based SDT should use the non-CDMI content type operations, for efficiency and backwards-compatibility reasons.

6.2. CDMI Features and SDT

CDMI provides a broad range of feature for Cloud Data Management, such as:

- discovering capabilities of a cloud storage provider;
- creating a new container;
- creating a new data object;
- listing the contents of a container;
- reading the contents of a data object;
- reading the value of a data object; and
- deleting a data object.

Moreover, CDMI provides a set of administrative operations, such as:

- managing domain objects representing the concept of administrative ownership (CDMI supports a hierarchy of domains and provides operations to manage those);

- queue object resource operations, providing first-in, first-out access for storing and retrieving data;

- capability query operations, allowing a client to find out about the subset of CDMI features that a server supports;

- exporting (and configuring the exporting of) data objects to other protocol domains such as NFS, iSCSI, WebDAV etc.;

- serialization and de-serialization of data;

- configure access control through ACLs;

- retention and hold management;

- scope specifications to allow clients to select data objects based on filter/search expressions;

- results specifications (to enable a client to specify subsets of data objects to be returned);

- logging;

- notification queues (for example for notfying clients about changes to a file system or to certain objects); and

- query queues (enabling clients to requests data objects based on meta data or content search expressions).

SDT over CDMI should specify a subset of these features and use the CDMI capability description mechanism to describe the subset of supported features.

6.3. CDMI Containers

Containers are a fundamental concept for CDMI, and they are used for grouping objects. In fact, containers are CDMI objects, and they can be addressed and manipulated using the same CDMI operations that are used for data objects.

With a flat naming scheme (as we expect DECADE to employ) there is no strong need for grouping objects in containers, so we recommend that the containers and container names should not be used for generating DECADE object names.

6.4. Object Identifiers in CDMI

CDMI required globally unique object IDs be used for all objects stored on a CDMI server, which is conceptually similar to the DECADE architecture requirements for naming.

In CDMI, objects are either accessible by their container-based hierarchical named such as "http://decade.example.com/root/vod/video1" or by their object ID such as "http://decade.example.com/root/cdmi_objectid/647284746393", with "647284746393" being the object ID.

CDMI specifies how object IDs should be generated. Object ID are variable length byte sequences with a maximum length of 40 bytes, and they provide the following structure:

```
___________________________________________________________
|   0    | 1 | 2 | 3|   4    |  5   |6|7| 8| 9|10|..|38|39|
|Reserved|Enterprise|Reserved|Length|CRC| opaque data     |
| (zero) |  Number  | (zero) |      |   |                 |
-----------------------------------------------------------
```

Figure 10: CDMI Object ID structure

Although CDMI Objects IDs could provide content hashes (in the opaque data fields), these IDs are not directly compatible to the current NI URI format. It is possible to convey the additional information of CDMI IDs in NI URIs, employing the extension mechanismsm, but syntactically, the NI URI would be different.

Although applications can treat these IDs as opaque bit strings, the format enables integrity checking for those applications that need it. In CDMI, the assumption is that the *server* generate these IDs, for example upon having received the object from a client over the upload interface. This server-based ID generation is the direct opposite of the client-based ID generation that we expect for DECADE.

6.5. Recommendations for SDT over CDMI

REC_19: The difference between CDMI's object ID syntax and the NI URI syntax should be addressed by either adapting CDMI's syntax or by defining a bijective mapping between CDMI and NI URIs.

REC_20: CDMI containers should not be used.

REC_21: CDMI should only by used in the non-CDMI content type mode

Figure 11: Recommendations

7. Security Considerations

Several security considerations need to be investigated for a DECADE SDT protocol and for DECADE in general. First, proper access control to objects stored at DECADE servers must be provided (OAuth is a means to do this, but the specific security implications for using OAuth in the context of DECADE need to be considered, and potential attacks need to be analyzed and described). Second, the potential for Denial-of-Service attacks on DECADE servers must be minimized. Finally, the integrity of data items stored at DECADE servers must be maintained, and clients must have a way to verify the integrity of data items they retrieve from a DECADE server (hash-based or self-certifying schemes as a component of the DECADE name space can be a means to provide these requirements, but the specific implications and potential attacks on data integrity need to be condidered carefully and described in detail). Future versions of this document will study these security aspects in more detail.

Also, SDT over HTTP-based protocols MUST support HTTPS. How applications choose whether to use HTTP or HTTPS will be discussed in a future version of this document.

8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3744] Clemm, G., Reschke, J., Sedlar, E., and J. Whitehead, "Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol", RFC 3744, May 2004.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[RFC5849] Hammer-Lahav, E., "The OAuth 1.0 Protocol", RFC 5849, April 2010.

[refs.CDMI] "CDMI", HTML http://www.snia.org/cdmi, September 2011.

[refs.decadearch] Alimi, R., Yang, Y., Rahman, A., Kutscher, D., and HP.

Liu, "DECADE Architecture", draft-ietf-decade-arch-03 (work in progress), September 2011.

[refs.ni-core]
Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., and P. Hallam-Baker, "The Named Information (ni) URI Scheme: Core Syntax", draft-farrell-decade-ni-00 (work in progress), October 2011.

Appendix A. Acknowledgments

Dirk Kutscher is partially supported by the SAIL project (Scalable and Adaptive Internet Solutions, http://www.sail-project.eu), a research project supported by the European Commission under its 7th Framework Program (contract no. 257448).

Jan Seedorf and Martin Stiemerling are partially supported by the COAST project (COntent Aware Searching, retrieval and sTreaming, http://www.coast-fp7.eu), a research project supported by the European Commission under its 7th Framework Program (contract no. 248036).

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SAIL or the COAST project or the European Commission.

Authors' Addresses

Dirk Kutscher
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: kutscher@neclab.eu
URI: http://dirk-kutscher.info/

Martin Stiemerling
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: martin.stiemerling@neclab.eu
URI: http://ietf.stiemerling.org

Jan Seedorf
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: seedorf@neclab.eu