# PEER Storage Engine Protocol

This is meant to be a technical design for PEER to be used by implementors. It is meant to be read together with the PEER Architecture. This is still work in progress to be discussed with the implementation teams before it is finalized.

## Notation

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 RFC2119.

## Introduction

PEER is described in the PEER Architecture. An instance of PEER consists of a user-interface that communicates with a back-end storage engine. The user-interface uses the storage engine to store and retrieve SAML metadata. The protocol used to communicate with the storage engine is described in the next section.

## Protocol Description

The PEER Storage Engine Protocol is a protocol for accessing a PEER storage engine (abbreviated "storage engine"). The responsibility of the storage engine is to provide a store and retrieve subsystem for SAML metadata. The storage engine MUST expose a WebDAV interface which SHOULD support https and at least BASIC authentication for WebDAV clients. A storage engine client is a WebDAV client which conforms to this specification. The storage engine protocol is then a profile of WebDAV.

A PEER storage engine client (abbreviated "client") MUST store each EntityDescriptor element as a separate resource (in the sense of RFC4918) inside a single collection (in the sense of RFC4918). The name of each resource MUST be a hash of the @entityID attribute of the EntityDescriptor element stored in the resource. Such a resource is called a "SAML metadata resource". A client SHOULD allow the hash algorithm to be configurable but MUST support at least SHA-1.

For more information on naming EntityDescriptor resources using a hash of the @entityID attribute see draft-lajoie-md-query-00 (this is a non-normative reference).

### Supported operations

A storage engine MUST minimally support the following operations from RFC4918:

- PUT
- DELETE
- GET, HEAD

### Cross-cutting concerns for operations

A client SHOULD ensure that updates are done as atomic operations possibly by establishing a lock (if the engine supports locking) either the collection or the individual resource before performing an update. A storage engine MUST permit configurable validation of SAML metadata and MUST apply any configured validation rule upon each update operation and MUST NOT allow the operation to success unless validation is successful. At minimum the storage engine MUST support a "schema-valid" validation rule which MUST ensure that provided SAML metadata is valid according to a predefined set of schemas. The storage engine SHOULD allow this set of schema to be configurable.

### Error handling

When a storage engine receives an update of a resource that does not pass the configured validation rules the storage engine MUST respond by returning a 409 (Conflict) and include a DAV:error element containing a validation-error element:

```
<!ELEMENT validation-error>
<!ATTLIST validation-error row CDATA #IMPLIED>
<!ATTLIST validation-error column CDATA #IMPLIED>
<!ATTLIST validation-error message CDATA #REQUIRED>
```

Example

```
<validation-error row="3" column="44" message="missing namespace declaration"/>
```

The @row and @column attribute SHOULD be provided (if applicable) to indicate where in the xml the validation error occured. The @message attribute MUST contain a human-readable description of the error.