Design - Hash URI Fragments

Hash-URI fragments do not have well-defined semantics. In HTML, they are typically used to take the user to a particular point on the page, and are also sometimes used to record application state. In XML and semantic web use, they are used as identifiers that are distinct from the base resource (i.e., without the hash URI fragment), but often contained in the same document (e.g., an RDF/XML element with the corresponding id attribute). In Fedora, they are intended to be descriptions that are distinct from, but dependent on, the base resource.

Expected Behavior

1. Triples whose subject is a hash URI can be added or removed via standard Fedora CRUD operations (POST, PATCH or PUT) on the resource from which the hash URI is derived
    a. When a resource is updated to include properties on hash URI fragments, they are automatically created.
2. Triples whose object is a hash URI can be added or removed via standard Fedora CRUD operations (POST, PATCH or PUT)
3. The RDF representation of a repository resource includes all triples from hash fragments associated with it. That is to say, all triples whose subject is a hash URI derived from that resource's URI
4. Any repository resource may link to any hash URI regardless of the presence of any triples associated with the hash fragment
5. A hash fragment should automatically be deleted when all of the triples that have that fragment as their subject have been deleted
6. Hash fragments should be permitted to include zero or more '/' characters

Current implementation - as JCR Nodes

In the current implementation, hash URI fragments are stored on JCR nodes under a special node named #. So properties about the hash URI fragment `/foo#bar` would be attached to the node `/foo/#/bar`.

1. Good - Properties on hash URI fragments can be added and deleted by updating the base resource (POSTing/PUTting RDF, SPARQL Update).
2. Bad - References to hash URI fragments are treated the same as references to base resources: they can only be made to existing fragments, and they are automatically removed when the fragment is deleted.
3. Good - When a resource is updated to include properties on hash URI fragments, they are automatically created.
4. Bad - The only way to delete a hash URI fragment is to delete its base resource – though there is a proposal to address this by automatically deleting them if they have no properties and/or inbound references.
5. Bad - Hash URIs also may not contain the "/" character, which is different than what RFC 3986 describes as possible with fragment URIs. The current behavior generates a stacktrace with this message: Hash URI resource created with too much hierarchy

Alternate implementation - as Property/Properties

An alternate implementation would be to store hash URI fragment properties in a JCR property. So the properties about all hash URI fragments would be stored as serialized RDF in the value of a property ?.

1. Properties on hash URI fragments would still be added and deleted by updating the base resource.
2. References to hash URI fragments would not be subject to the same restrictions as references to nodes, and would not be automatically removed when they were deleted.
3. Hash URI fragments would not have any persistence other than triples in the serialized RDF, so adding triples would effectively create them, and deleting the triples would effectively delete them.
4. As properties, hashURIs could contain the "/" character, as described by RFC 3986.

Questions

1. One of the driving use cases for hash fragments is proxies and ordering. What, if any, impact would a properties-based implementation have on that use case?
2. Would properties stored in a properties-based implementation continue to have all the restrictions placed on properties on nodes? E.g., could a property refer to a non-existent repository resource? Would a property that referred to a repository resource be deleted when that resource was deleted?