<table>
<thead>
<tr>
<th>CONTENTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>2</td>
</tr>
<tr>
<td>Namespaces</td>
<td>4</td>
</tr>
<tr>
<td>Motivation</td>
<td>4</td>
</tr>
<tr>
<td>Proposed Changes</td>
<td>5</td>
</tr>
</tbody>
</table>
Change Summary

Version 0.1 (Jens Hübel, Florian Müller)

- Initial draft

Version 0.2 (Jens Hübel, Florian Müller, Julian Reschke)

- Clarification prefix/namespaces
- Added URIs to property definitions
NAMESPACES

MOTIVATION

Version 0.5 of the CMIS specification handles properties as a flat list of key-value pairs. That may cause name conflicts between CMIS base property names and type specific property names. The same holds true for type names. It is likely that some repositories already use the type names “Document” or “Folder”.

During the face-to-face meetings and in various discussions proposals have come up to extend this naming to a more powerful mechanism by introducing namespaces. The following proposal is an approach to extend the CMIS specification with a simple concept of namespaces, being widely compatible with the existing specification. The proposal touches type names and type query names as well as property names and property query names.

Furthermore, many other systems use globally unique property identifiers in order to promote distributed extensibility. Examples are RDF (which uses URIs) and JCR and WebDAV (which use XML expanded names, consisting of a namespace URI and a local name). If these types of systems are accessed over CMIS, it is desirable to expose the “native” property identifier to clients. On the other hand, when CMIS based systems are accessed through JCR or over WebDAV, it would be useful if the repository can expose a globally unique identifier as well.

Therefore, we propose to make that globally unique property name an optional field of the property definition.

Introducing namespaces provides the following advantages:

1. The CMIS specification defines a set of standard properties. An existing repository may have its own properties with the same names but with different semantics or different sets of constraints. A namespace allows clearly distinguishing between CMIS properties and repository or type specific properties.
2. Certain repositories allow using the same name of a property to be used at more than one place with a different meaning. Common property names like “Name” or “Invoice” might exist in different flavors and in different application contexts.
3. Future versions of the CMIS specifications may define additional standard properties (e.g. for Records Management). By introducing a separate namespace for the CMIS standard we can guarantee backwards compatibility between a newer version of the specification and an existing repository or application implementation.
4. A namespace can be the foundation to introduce extended properties (e.g. hierarchies). Here are some relationships to other upcoming proposals, for example Aspects/Mixins that should be synchronized.

A namespace is used to group properties within a type. A namespace is expressed by a prefix to the property name. A prefix on the object type level helps to distinguish the CMIS “Document” type from a repository “Document” type.

Using prefixes is optional with the exception of the CMIS standard properties and types. A repository that does not support namespaces can provide the same implementation as for version 0.5 of the CMIS spec and is still CMIS conformant.
**PROPOSED CHANGES**

Add section Part I, Section Content / Data Model / Namespaces

Prefixes are identifiers according to the SQL 92 rules for identifiers. Prefixes can be used to structure type query names and property query names and to ease providing unique identifications. Prefixes are separated from the type query name, property query name or other prefixes by a colon (":") and can be hierarchical. Examples are “myrepository:name” or “MyRepository:Finance:Order:OrderNumber”. Using a namespace is optional. A prefix on a property name or type query name is only provided by the repository and opaque to a client. A CMIS client MUST NOT interpret namespaces. Malformed prefixes (like providing a wrong prefix in a property or type name) are handled by the repository in the same way as providing an invalid property or type name. Namespaces SHOULD be unique within a repository. A query containing properties and/or types with prefixes still has to conform to the SQL92 standard.

The prefix “cmis:” identifies a reserved namespace and MUST NOT be used by any repository except for the names that are defined in this specification.

The length of type query name and property query name plus prefix is not limited to 128 characters as in SQL-92.

Part I, Section Content / Data Model / Object last paragraph, after second sentence “Within an object, each property is uniquely identified by its name.” add the following:

A property name can have a prefix. A prefix is separated from the property name by a colon (":"). A colon can either separate a prefix from a property name or a prefix from a prefix. The local part (behind the last colon reading from left-to-right) is always a name. Using prefixes is optional and implementation specific.

Part I: Replace all predefined property names so that they are prefixed with “cmis:”, e.g. change “CreatedBy” to “cmis:CreatedBy”

Add the prefix “cmis:” to the values of ObjectTypeQueryName and RootTypeQueryName of the object types “Document”, “Folder”, “Relationship” and “Policy”.

Part I, Section Content / Data Model /Query

Change the following lines in the BNF of the CMIS query grammar from:

```plaintext
<correlation name> ::= <identifier>
<table name> ::= <identifier>
<column name> ::= <identifier>
```

to:

```plaintext
<correlation name> ::= < identifier with namespace >
<table name> ::= < identifier with namespace >
<column name> ::= < identifier with namespace >
```

Add line:

```plaintext
<identifier with namespace> ::= [({<identifier> ":"}...]) <identifier>
```
unchanged:
<identifier> ::= !! As defined by SQL-92 grammar

Proposed changes for globally unique property id:

Part I, Section 2.7.3.2:

Add the following attribute:

Name: globallyUniqueName
Type: string
Function: A globally unique name for this property. This is either a URI (RFC 3986, Section 3), or an XML NS Expanded Name (XMLNS, Section 2.1), consisting of a namespace URI and a local name. The lexical representation for the first form is just the URI, for the second form it is the Clark Notation ("{" + namespaceuri + "}" + localname) of the expanded name.

Open Issues:

1. Are both forms required? Or should we just promote URIs, which may cause problems for existing expanded names that do not have an equivalent URI?
2. Use IRI (RFC 3987) instead of URI?
3. Require that the globallyUniqueName is returned as additional identifier on the wire (Atom)?