Summary of Content Management Interoperability Services (CMIS) 1.0

Today's business contents are "trapped" in disparate content management systems. It is difficult for an enterprise to integrate contents that are stored in different repositories, and it is expensive for a vendor to develop and maintain application that can access different repositories. To mitigate this "content silo" problem, CMIS specifies an interoperability interface for content management systems so that generic applications can be developed that can access any CMIS-compliant repository.

CMIS is a protocol-layer interface. But it is not singularly defined for a specific network protocol. The CMIS specification contains a domain model and a set of protocol bindings. For Version 1.0, two protocol bindings are defined: a WSDL-based Web Services binding (service-oriented), and a "RESTful" Atom Publishing Protocol binding (resource-oriented).

The CMIS interface is designed to be layered on top of existing content management systems and their existing programmatic interfaces. It is not intended to prescribe how specific features should be implemented within those systems, nor to exhaustively expose all of a system's capabilities through the CMIS interface. Rather, it is intended to define a generic/universal set of capabilities provided by a content management system and a set of services for working with these capabilities. A few CMIS capabilities are optional, primarily because they are not supported by all existing systems. CMIS provides a service to allow an application to discover programmatically whether or not each optional capability is supported by a compliant repository.

The CMIS interface exposes multiple repositories that are accessible at a given service endpoint. However, aside from listing the accessible repositories, all other services provided by CMIS 1.0 are scoped to a single repository. Within a repository, CMIS defines a model of typed objects, with type hierarchy and type inheritance (single inheritance). Services are provided for an application to discover object type definitions. However, object type management services (e.g., to create, update, or delete an object type) are outside the scope of Version 1.0. Each CMIS object has a repository-assigned unique identifier and a set of properties (metadata), and may have a content stream (digital asset) and renditions (alternate representations such as a thumbnail). What properties an object has, and whether or not it can have a content stream, are defined by its object type. An object type definition contains a list of object type attributes (such as type name and parent type) and a list of property definitions. Each property definition specifies the attributes of a property, such as its name, data type, whether it is single-valued or multi-valued, etc.

Four kinds of objects are defined: Document, Folder, Relationship, and Policy. They are defined as four separate root types in the type hierarchy, without imposing any affinity among them to allow maximum compliance. A repository may further define subtypes of these root types in support of specific applications. A subtype inherits property definitions from its supertype and may have additional property definitions of its own. Basic CRUD services (Create, Retrieve, Update, Delete) are provided for these objects and for content stream. In addition, a query service is provided for finding objects.

- A Document object represents an information asset managed by the repository. It may have a content stream, can be versioned, and is queryable. A Document may be filed in zero or more Folders. Services are provided for creating a new version of a Document (checkout, checkin), and for retrieving all versions of a Document. A linear (chronological) version history is supported.

- A Folder object represents a logical container that may contain Documents and Folders as its child objects. It does not have a content stream and cannot be versioned, but it is queryable. The collection of all the Folders in a repository forms a strict hierarchy. Services are provided for navigation through this Folder hierarchy, and for filing/unfiling a Document or moving a Folder sub-tree from one location to another in the hierarchy.

- A Relationship object represents a binary, peer-to-peer, directional relationship between two other objects. It does not have a content stream, cannot be versioned, and is not queryable. However, services are provided for navigation through Relationships.

- A Policy object represents an administrative policy that can be applied to, or removed from, another object.

For the query service, CMIS supports a type-based, structured query language and leverages the SQL language. Specifically, a relational view is implicitly defined on top of the CMIS data model: a virtual table corresponding to each object type, and a table column corresponding to each property of an object type. Through this view, SQL-like queries may be used to find objects. The query language borrows a subset of the SQL-92 SELECT statement grammar, both syntax and semantics, plus the following extensions to facilitate content management:

- Support for multi-valued properties,
- Support for full-text search,
- Support for narrowing the search scope to a folder or a folder sub-tree.

CMIS provides an ACL-based access control capability, allowing an application to get, set, or alter simple access control rules for an object. An application may use or request either repository-specific permissions or CMIS-defined permissions (“read”, “write”, and “all”). For the latter, a best-effort mapping to a repository’s native permissions is expected.

CMIS also provides a “change log” capability to allow an application (such as a crawler) to discover the set of changes that have occurred to objects stored in the repository since a previous point in time.

The AtomPub binding is based upon the Atom Syndication Format (RFC 4287) and the Atom Publishing Protocol (RFC 5023). It supports a RESTful architecture style of computing. An implementation of this binding must be compliant with RFC 4287 and RFC 5023. In this binding, a client interacts with the repository by acquiring an AtomPub service document using a URI supplied by the repository provider. The client will then choose a CMIS collection, and start accessing the repository by following the references in the returned documents. User authentication is handled by the transport protocol.

This binding consists of an AtomPub service document specifying CMIS service collections, atom collections, feed and entry documents. CMIS extends the Atom and AtomPub documents utilizing the Atom and AtomPub extension mechanism. CMIS also leverages link tags to specify additional resources related to the requested resource.

- Namespace: A CMIS-Core namespace (cmis:) and a CMIS-RestAtom namespace (cmisra:) are used in addition to the Atom (atom:) and AtomPub (app:) namespaces.

- Media type: The following media types are used in addition to the AtomPub Service, Atom Feed, and Atom Entry media types:
  - CMIS Query
  - CMIS AllowableActions
  - CMS ACL
  - Atom Document (Entry or Feed) with CMIS Markup
  - Atom Feed Document with CMIS Hierarchy Extensions

- Link relation: A number of CMIS-specific link relations are used in addition to IANA-registered Atom link relations and link relations proposed by the Hierarchy Navigation Internet Draft and the Versioning Internet Draft.

- Collection: CMIS collections are defined for folder children, relationships, and policies. In addition, several service collections are defined: root folder collection, query collection, checked out collection, unfiled collection, and types children collection.

- Feed: Atom feed documents are defined for object parents, folder descendants, folder tree, all versions, changes, and type descendants.

- Resource: Atom entry documents are defined for CMIS objects and type definition. A content stream is referenced by the “src” attribute of Atom content as well as by the “edit-media” link relation. Each rendition is referenced by an “alternate” link relation.

- Exceptions are mapped to appropriate HTTP status codes.

The Web Services binding is defined by WSDL, which supports, one-to-one, the services defined by the CMIS domain model. Two XSD schemas are used by the WSDL. One defines data types and the other defines message formats. All types support xs:any for holding repository-specific information. All input messages and some output messages have an “extension” element for passing repository-specific information. All services must report errors via SOAP faults. All service endpoints must be MTOM enabled. Content range-retrieval (by offset and length) is supported. A Web Services binding implementation must comply with WS-I Basic Profile 1.1 and Basic Security Profile 1.0. For user authentication, it should support WS-Security 1.1 for Username Token Profile 1.1.

To validate the specification, a number of CMIS TC members have built prototypes, either to provide or to consume CMIS services, and tested interoperability between different vendor implementations.