EMC²
where information lives®

White Paper

Content Management Interoperability Services: Defining Web Services for Sharing Information among Disparate Repositories

Technology Concepts and Business Considerations

Abstract

EMC, IBM, and Microsoft have drafted Content Management Interoperability Services (CMIS), a proposed standard for a set of web services for sharing information among disparate content repositories. CMIS is designed to solve a difficult business problem—ensuring interoperability for people and applications using multiple content repositories. This white paper describes the technical capabilities of CMIS and identifies how the proposed standard, once adopted and implemented within enterprise applications, will accelerate the development of interoperable content solutions.

September 2008

h3951

Table of Contents

Executive summary

EMC, IBM, and Microsoft have drafted Content Management Interoperability Services (CMIS), a proposed standard for a set of web services for sharing information among disparate content repositories. This proposed standard seeks to ensure interoperability for people and applications using multiple content repositories. The three companies have jointly submitted CMIS to the Organization for the Advancement of Structured Information Standards (OASIS), to initiate the technical development process that leads to standardization.

CMIS is designed to augment existing enterprise content management (ECM) systems and their current application interfaces. CMIS focuses on the basic content capabilities of an ECM system—the create, read, write, delete, and query functions. CMIS defines the basic content services and a service-oriented interface for working with these capabilities. CMIS incorporates contemporary concepts of a web services orientation and specifies both SOAP-based (Simple Object Access Protocol) and REST-based (Representational State Transfer) protocol bindings.

Introduction

This white paper describes the capabilities of Content Management Interoperability Services (CMIS), a proposed standard for a set of web services for ensuring interoperability among disparate content repositories. CMIS promises to transform how knowledge workers and applications access and manage content among disparate repositories. CMIS standardizes the basic operations of an ECM system, and make them widely available as web services. As it gains popularity among application developers, CMIS has the potential to fundamentally change how content is managed and consumed, both within a corporate intranet and over the public Internet.

Audience

This white paper is intended for experienced application developers and IT managers who want an overview of the proposed Content Management Interoperability Services (CMIS) standard, including its benefits and sample use cases.

The case for content interoperability

An initiative for standardizing basic content services

Working together to solve a difficult business problem, EMC, IBM, and Microsoft have drafted Content Management Interoperability Services (CMIS), a proposed standard for a set of web services for sharing information among disparate content repositories. This proposed standard is designed to ensure interoperability for people and applications using multiple content repositories. EMC, IBM, and Microsoft have submitted CMIS to the Organization for the Advancement of Structured Information Standards (OASIS), to initiate the technical development process that leads to standardization. The three vendors will continue to spearhead efforts within a new OASIS technical committee to build broad industry support for this initiative.

It is important to emphasize what CMIS seeks to accomplish and what is beyond the scope of this multi-vendor initiative. CMIS is designed to augment existing enterprise content management (ECM) systems and their current application interfaces. CMIS focuses on the basic content capabilities of an ECM system—the create, read, write, delete, and query functions. When deployed, CMIS ensures interoperability by defining how these core content management capabilities function in a uniform manner over a variety of

ECM systems. Beyond this limited (albeit important) set of operations, CMIS does not seek to standardize other capabilities of a content repository. Nor does CMIS seek to standardize the administrative functions supported by each ECM system.

Beyond information stovepipes

Absent CMIS, companies face many challenges as they enhance their enterprise information infrastructure and leverage their IT investments. More often than not, different business units within a firm manage content using multiple content repositories. For instance, the technical documentation group may rely on one ECM system, the product engineering team a second, the field sales and support organization a third, and the corporate marketing program a fourth. A company may have deployed this kind of heterogeneous environment for any number of reasons. Business units often operate separately and adopt different ECM systems that are designed to solve particular problems. As another example, a company that grows through acquisitions often inherits disparate systems in the process. Consolidating them into a single application environment is often disruptive and costly, and may not deliver the most optimum solution. Thus a company needs to find a way to access and share content among the multiple systems.

The net effect is stovepiped information access and delivery. Each team or business unit systematically manages content within its own environment—difficulties often arise when trying to work across system-defined boundaries. For example, technical writers might use one ECM system to produce product documentation. This system works well when writers just need to create, review, and update technical manuals and online help files. Yet all too frequently, writers need to access the technical specifications, maintained by engineers who are using a different ECM system. Writers have to overcome the barriers of disparate content repositories to get the information they need to do their work.

Without an accepted industry standard, application developers cannot easily develop new applications for disparate content repositories. Rather, each ECM system exposes its own APIs and services for accessing and managing the underlying repository. Invariably, application developers build and extend separate yet incompatible ecosystems around each repository. The end results are the unintended consequences of a balkanized information infrastructure—discrete environments, multiple investments, added costs, a loss of business flexibility and an inability for content to easily flow across multiple business units.

Business benefits

Needed are web services for people and applications to interoperate with content managed among multiple repositories without requiring a tight coupling between a client and each repository it accesses. CMIS promises to simplify integration challenges and help provide three business benefits.

- Knowledge workers can use a single application to interact with content stored and managed by several ECM systems. End users no longer need to use a unique application (based on a separate interactive experience) for managing each content repository.
- A company can deploy an enterprise workflow that interacts with content managed by multiple ECM systems. The firm can create and manage business processes that transcend content repositories in a standardized fashion.
- Application developers can rely on a standardized web-based, service-oriented interface to develop their applications once, and deploy them across multiple ECM systems. They then have the flexibility to quickly respond to new operational requirements. They can leverage a web services infrastructure to extend the reach of their distributed applications. They can easily modify the application once and continue to manage content across multiple repositories.

In short, CMIS provides the basic content services for connecting to heterogeneous repositories within a corporate intranet or over the public Internet.

Alternative approaches for content interoperability

Of course, CMIS is hardly the first initiative that seeks to standardize interoperability among disparate content repositories. However, it is the most focused effort to date for building upon an existing enterprise infrastructure, and for using web services to support loosely coupled application integration techniques.

Minus suitable standards, interoperability among multiple ECM systems has been difficult, costly, and time consuming to implement. Standards such as JCR, WebDAV and Atom do approach these challenges, yet each is different from CMIS in some very significant ways.

JCR (JSR-170/JSR-283)

As the specification for a Java platform API for accessing content repositories in a uniform manner, Java Content Repository (JCR) focuses on a comparable objective to CMIS. But the approach that JCR takes inhibits a wide deployment across the Internet and other heterogeneous, distributed computing environments.

- The JCR standard is a Java-language only standard. By comparison, CMIS uses web services that are not specific to a particular programming language.
- The JCR standard requires a substantial implementation effort for existing ECM systems. JCR uses an abstract data model (with a strict hierarchy of typed nodes) that is different from those of widely deployed ECM repositories. To layer JCR, an existing repository has to simulate this rudimentary node hierarchy, along with all its operations, using high-level constructs supported by the repository.
- Furthermore, JCR is function-rich. While an ECM repository may support similar functions, any minor deviation in the behavior of any of these functions poses a compliance challenge because a deployed repository can not change its behavior freely. Therefore, JCR is not an interface that can be easily layered on top of an existing full-function repository. In contrast, CMIS restricts itself to specifying only generic, traditional concepts and basic functions that can be layered on most existing ECM implementations.
- The JCR standard defines a session-based API that is tightly coupled to a repository. This API is not intended for a Services Oriented Architecture where an application makes loose connection to a content repository.
- The JCR standard does not leverage or interact with the newer Web 2.0 technologies.

CMIS addresses situations where not all repositories or applications are implemented in Java, where some existing repositories cannot support JCR semantics, and where loosely coupled integration techniques, Service-Oriented Architecture, and Web 2.0 technologies are appropriate for an enterprise.

WebDAV

WebDAV is an early standard that extends the HTTP 1.1 protocol to support distributed authoring on the Web. As such, it offers a basic level of content interoperability for web resources. WebDAV defines a simple property bag model for web resources, and supports resource collection through hierarchical naming in a file-system fashion.

While this simple model provides the flexibility needed for various unstructured applications (such as web authoring), it does not provide sufficient discipline (such as object typing and schema) to support enterprise applications. Furthermore, the services provided by WebDAV are limited to a handful of HTTP methods (including the WebDAV extension). An ECM repository typically offers a broader set of services, such as navigating folder hierarchy, query, and discovering type definitions.

It is not wise to load (extend) the simple and elegant HTTP protocol with a large set of domain-specific methods along with an extensive data model. Moreover, since WebDAV is explicitly tied to HTTP,

interoperability ends wherever other messaging services or protocols are used within an enterprise environment (such as Java Messaging Service (JMS)).

Atom Publishing Protocol

Atom Publishing Protocol (APP) is an HTTP-based protocol for publishing and updating web resources. Together with the Atom Syndication Format (ASF), it addresses content interoperability, particularly for blogs and news feeds, and has become a popular element in many Web 2.0-style solutions.

Like WebDAV, APP is also tied to HTTP. But the two protocols, working together, accomplish much more. APP/ASF defines a data model that is even simpler than the WebDAV data model. It consists of entry and collection of entries (feed), but not nested collections (hierarchy of collections). A set of metadata properties are specified, which is intended for authored content and can be a bit awkward for other content.

While APP is becoming a popular interface for accessing web resources (delivering content to an interactive client), it does not have sufficient modeling capability for managing enterprise content. CMIS defines a model for enterprise content, and leverages APP as one of the protocols an application may use to deliver content.

The promise of CMIS

Thus, CMIS seeks to become a next generation standard for powering integrated content applications over the Web. CMIS defines the operations for a core set of capabilities provided by ECM systems. It defines the basic content services and a service-oriented interface for working with these capabilities. CMIS is designed to ensure interoperability by providing uniform access to content among disparate repositories.

CMIS incorporates contemporary concepts of a web services orientation. Specifically, CMIS is designed to work with existing content repositories that need to support interoperability features without modifying their underlying implementations. CMIS supports the development of composite applications, those rapidly constructed of loosely coupled services. Moreover, CMIS is designed to support content interoperability across an extended enterprise. These web services can be distributed beyond the boundaries of an enterprise, and can facilitate the development of applications that work at the scale of the Internet.

Making it easy to implement interoperable repositories

Web services for enterprise content management

CMIS defines the core capabilities for implementing interoperable ECM repositories. These capabilities are designed as simple, generic functions for managing content regardless of the underlying platform or storage mechanism.

CMIS connects disparate repositories through a service-oriented interface, as shown in Figure 1. CMIS is implemented by each ECM system to work with the content and metadata within its repository in a standardized fashion.

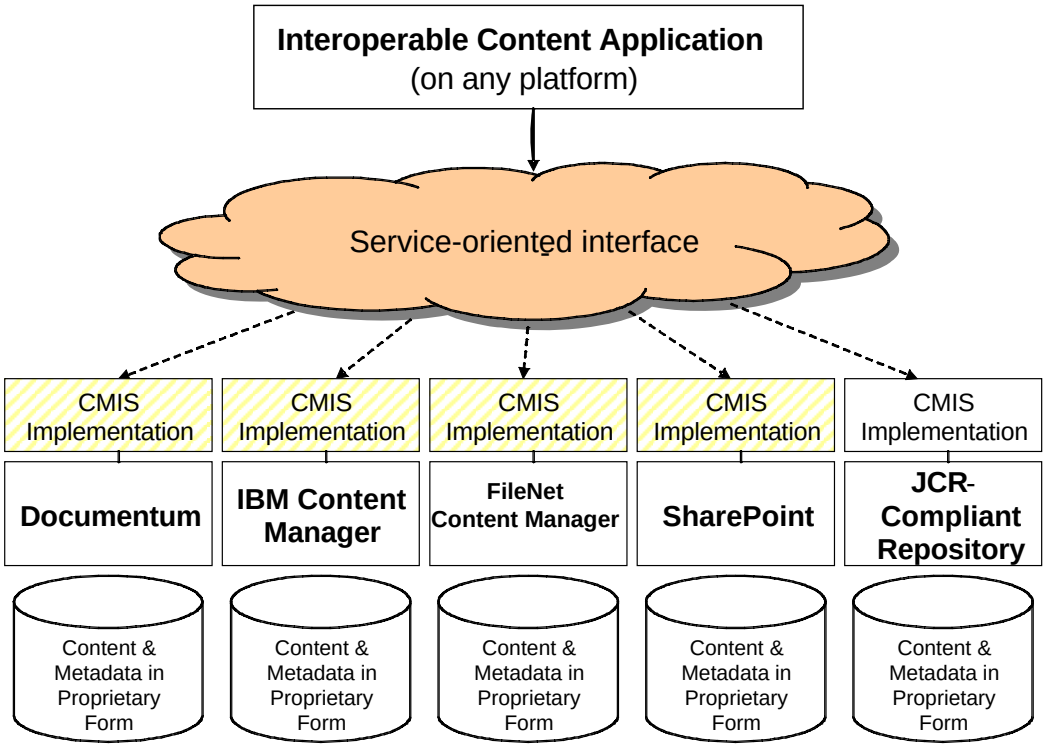

Figure 1. CMIS relies on a service-oriented interface to provide connections to disparate content reportrices

CMIS exposes services for:

- Discovering object type definitions and other repository information (including the optional capabilities that are supported by a particular repository)
- Creating, reading, updating, relating, and deleting objects
- Filing documents into one or more folders (if the repository supports the optional multi-filing capability)
- Navigating and traversing the hierarchy of folders in a repository
- Creating and accessing versions of documents
- Querying a repository to retrieve one or more objects matching user-specified search criteria, including full-text search

CMIS defines functions as language-independent services that are not limited to a single programming language or a fixed transport protocol. CMIS currently specifies two separate protocol bindings, including one that is SOAP-based (Simple Object Access Protocol) and another that is REST-based (Representational State Transfer). CMIS provides a lightweight, loosely coupled interface to a repository, independent of the underlying platform and transport protocol.

CMIS comprises an elegant yet simple approach to content interoperability, one that operates within an existing heterogeneous enterprise infrastructure. CMIS is a limited yet complete interface. It does not seek to standardize all of the functions available within an ECM system. CMIS is not designed to be a substitute for a repository's native APIs. It does not require organizations to make modifications to their existing ECM solutions, applications, or platforms.

Let us examine how CMIS can function within a distributed environment and interoperate with content managed within disparate repositories.

Defining objects and actions for a familiar file cabinet metaphor

CMIS defines a set of objects that are used to implement a familiar file cabinet metaphor for managing content among disparate repositories. Three kinds of objects are stored within a CMIS-compliant repository—documents, folders, and relationships.

- Documents represent individual content objects stored within a repository. A document may or may not include a content stream, the bytes that comprise the raw digital assets (such as a word-processing document or a JPEG image).
- Folders represent organizational containers in which documents or other folders are grouped.
- Relationships represent an explicit directional reference (or binary relationship) between exactly two documents or folders in the repository.

CMIS does not try to standardize the underlying security model. Rather, CMIS simply leverages the security capabilities provided by each repository. CMIS thus relies on the authorization, authentication, and access control functions that are defined by each ECM system.

Object identity

Each object has an opaque and non-updatable Object Identity (OID), which is assigned by the repository when the object is created in the repository. An OID uniquely identifies an object within a repository regardless of the type of object. OIDs are permanent—they remain unchanged during the object's lifespan and are not to be reused/reassigned after the objects are deleted from the repository. CMIS can always identify any object stored within a compliant repository.

In addition to an OID, a repository may assign a Uniform Resource Identifier (URI) to an object, enabling an application to access the object as a web resource using standard web-based protocols and tools. However, the permanency of a URI is repository-specific and the validity of a URI is outside the scope of CMIS.

Object types

CMIS is based on the concept of object types. Each object type defines a property schema for objects of that type. Each object type also provides a search scope for queries. Type hierarchy and single inheritance are supported. That is:

- A subtype inherits its ancestor types' property definitions, and may have additional property definitions.
- Query scope is automatically expanded to include instances of all descendant types.

CMIS also supports the discovery of object types and the retrieval of type definitions. But object type management—the mechanisms for creating, modifying, and deleting object types—is not standardized by

CMIS. These management capabilities will continue to be performed through a repository's native interfaces.

Each object type defines its own set of properties. Each property is named and holds typed data values (including string, decimal, integer, Boolean, and date/time as specified by XML Schema). A property may be single-valued or multi-valued. A single-valued property contains a single data value, whereas a multi-valued property contains an ordered list of data values of the same type.

Folder hierarchy and document definition

CMIS supports a folder hierarchy for grouping documents. In particular, a folder:

- Has properties but no content stream
- Is queryable, but not versionable
- Has implicit containment relationships with other objects, where it can contain documents and folders
- Must be contained in another, one and only one, folder.

In comparison, a document is queryable and versionable. It can be contained in a single folder or optionally in any number of folders or not in any folder at all. Consequently, an application can always locate a document by accessing the folder (or folders) in which it is contained without knowing its OID. A service is also provided for retrieving documents that are not contained in any folder.

Versioning

CMIS supports versioning of document objects. A version of a document object is an explicit copy of the document object, preserving its state at a certain point in time. A check-out/check-in procedure is defined for creating a new version of document.

In general, a document object may have multiple versions co-existing in a repository. Each version is itself managed as a separate, standalone document object. As such, it has its own object identity and its own administrative metadata such as Creation Date.

Among all the versions of a document object, the one that has the most recent date/time is called the *Latest Version* of the object. Whether the non-latest versions of a document object are updatable, queryable or full-text-searchable is repository-specific. An explicit deletion of the latest version of a document causes a previous version, if there is one, to become the latest version. Whether this is considered a modification to that previous version is also repository-specific.

Basic operations—create, retrieve, update, delete

CMIS provides OID-based Create, Retrieve, Update, Delete (CRUD) services for objects. These actions define the fundamental operations that compliant repositories need to manage.

- The "Create" services create an object and returns an OID.
- The "Retrieve" services return the properties or content-stream of an object. These services may be invoked with a filter specifying the properties to be returned.
- The "Update" services update the properties or content stream of an object. A multi-valued property can only be updated by replacing the entire list of its values.
- The "Delete" services delete one or more objects. For versioned documents, all versions can be deleted.

Query capabilities

CMIS provides standard mechanisms to query disparate repositories and return results in a consistent fashion. An application can invoke queries that interrogate and display information from any CMIS compliant repository.

The query capabilities within CMIS (CMIS SQL) are based on an implicit relational view defined for the CMIS data model and a subset of SQL 92, a widely adopted and proven query language for relational databases. To this SQL-92 subset, the following extensions are added to meet content management needs.

- **Text search.** Text search standardizes the call interface into a repository and the presentation of the results. There is no attempt, however, to standardize the string matching or linguistic processing that a search engine performs within a repository. Consequently each repository continues to process the text query according to its own capabilities. CMIS simply standardizes the services for the interactions.
- **Multi-valued properties.** While SQL is limited to single-valued attributes, content queries often pertain to multiple valued properties, such as multiple authors for a document. CMIS SQL adds quantified predicates to filter multi-valued properties.
- **Scoping basis of a folder.** SQL per se cannot limit a query to a folder. CMIS SQL can relate query capabilities to folder constructs, and thus can limit the search scope of a query to documents and folders within a folder or a folder hierarchy.

These extensions are incorporated in such a way that the integrity (syntax, semantics, and BNF production rules) of the adopted subset of SQL is preserved. Accordingly, CMIS SQL is designed to maintain its compatibility with SQL, and reserve an option to incorporate a larger subset of SQL without conflict in the future. This compatibility also allows an enterprise to draw upon the skilled base of application developers and business analysts who have extensive experience implementing SQL-driven applications.

Protocol bindings

The CMIS data model and services are independent of the protocol that is used to invoke the services. Currently, it supports both SOAP-based and REST-based protocol bindings. These bindings are popular with application developers building distributed applications. Applications developers can choose the set that is best suited for their enterprise application and ecosystem. Additional protocol bindings may be supported by CMIS in the future.

SOAP

The SOAP-based binding defines an activity-oriented interface that exposes operations to which parameters, including in most cases an OID, are passed. Web service frameworks supporting SOAP and related standards (such as WS-Security) allow for a separation of business logic from messaging or other infrastructure concerns. Thus CMIS services should generally operate in any such environment. While the array of web services related standards is too vast and in flux to be applicable to CMIS in its entirety, Web Service Interoperability Basic Profile (WS-I BP) and Web Service Security (WS-Security) must be supported in compliant implementations.

The CMIS SOAP binding includes Web Services Description Language (WSDL) for the services. This WSDL may be used, for example, to generate code stubs in the programming language of the client. The WSDL includes XML schema that defines the message formats of the service as well as the data formats of the CMIS resources (document, folder, relationship). Of course, the SOAP interface may be invoked using a variety of transport protocols (that is, HTTP, JMS, and others).

Many enterprises have already invested heavily in SOAP-based web services ecosystems. Not only do they have SOAP experience in their IT and application development staff, they often also have extensive

development tooling and runtime ecosystems in place. These factors drove the inclusion of a CMIS SOAP binding in the draft specification.

REST

REST is an architecture style that is particularly popular with application developers building Rich Internet Applications (RIA). These applications, such as mashups, are characterized by implementations layered over the infrastructure of the world wide web RESTful[1] interfaces are oriented around resources, on which a limited set of operations may be executed. At present, all RESTful interfaces are invoked via the HTTP protocol and execute four operations—GET, PUT, POST and DELETE.[2]

The CMIS REST-based binding is implemented as an extension to the ASF/APP specifications. The resources against which services are invoked are the repository itself, the documents, folders and relationships, and collections of these objects. All CMIS operations can be instantiated by sending an HTTP GET, PUT, POST or DELETE to one of these resources with additional required values in the query string, the HTTP headers, and the HTTP payload.[3] Specifically, the APP/ASF data model is used as the content delivery model, whereas the CMIS data model is used as the persistence (repository) model.

While RESTful interfaces are dramatically increasing in popularity, such technologies are rather new relative to SOAP technologies and are thus lacking in development and deployment tools. Nevertheless, there are situations, such as Ajax-style programming, where the REST binding may be the most appropriate

Application scenarios: CMIS in operation

How does CMIS transform the ways in which companies can manage content across a distributed work environment, and add value to business operations? Let us consider two application scenarios that highlight the potential for CMIS in operation.

- The first is a mortgage processing application that incorporates content from multiple parties through a distributed, content-centric business process.

- The second is a customizable personal portal where knowledge workers rely on a browser-based wizard to select, access, and interact with content stored within multiple CMIS-compliant repositories.

These scenarios illustrate the business benefits of content interoperability. Application developers can cost-effectively implement scalable and extensible solutions, where they develop the application once and then manage content across multiple repositories. Once the new application is up and running, content flows as expected across their distributed work environment. As a result, knowledge workers experience a seamless flow of content that makes it easier for them to do their jobs.

---

[1] REST follows the designed principles presented by Roy Fielding, one of the principal authors of HTTP, in his doctoral dissertation. See: "Representational State Transfer (REST)," in *Architectural Styles and the Design of Network-based Software Architectures*, University of California at Irvine, 2000, (www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm).

[2] There are efforts within the industry underway to explore alternatives, such as RESTful SOAP-based services.

[3] Some services currently available on the web are called "REST services" but do not adhere to all of the principles outlined in Fielding's dissertation; instead they simply pass XML in HTTP requests. The CMIS binding, by comparison, aims to be purely RESTful and conform to those design principles.

With this added flexibility come new content applications. Companies further leverage their ECM investments by deploying a distributed, yet interoperable application infrastructure, based on web services. Thus the return on investment in ECM for a company—improved productivity, innovative and effective business processes, together with rapid access to actionable information—falls right to the bottom line.

Mortgage processing through an extended enterprise

NationalMort, a hypothetical mortgage company, serves residential and commercial customers across the United States. The company has developed an innovative, content-centric solution for processing and tracking mortgage applications. Although its business is national in scope, it works with a series of local business partners to obtain flood reports, credit reports, title reports, appraisals, and so forth, that are required to secure financing.

NationalMort has designed and implemented a business process for this extended enterprise, incorporating content from partner firms that operate at a local level. NationalMort can automatically request and obtain content from its business partners through a standard content interface (CMIS) without needing to know how the partner firms manage their own content. As a result, NationalMort can manage all of the content electronically, and can easily add new firms to its enterprise ecosystem.

By managing content in a systematic fashion, the company both reduces its cost of operations, ensures compliance with various regulatory mandates, and has the ability to rapidly and cost-effectively respond to requests for information that might arise in the course of doing business. With its embrace of web services and its use of CMIS, partner firms find that it is easy to do business with NationalMort—they can make their reports available to the company's required business process without needing to change their own ECM systems.

CMIS and a business process

Mortgage processing at NationalMort is a content-centric process that involves such things as loan applications, flood reports, credit reports, and other types of business objects. As multiple firms are involved in the mortgage processing business, all of the relevant applications and tools needed cannot be deployed within a single IT infrastructure. Rather the business process must facilitate the flow of content across the extended enterprise.

The simplified, sample business process is summarized in Figure 2. The filing of a loan application launches an instance of the business process. The process then branches into parallel tracks to "Request Flood Report," "Request Credit Report," and similar steps. Steps in these paths involve interactions with partner firms who then execute their own internal processes to generate requested reports and make them available within NationalMort's business process. Once all the requested documents are available, the "Final Review" step is performed by a loan officer and the mortgage is either approved or denied.

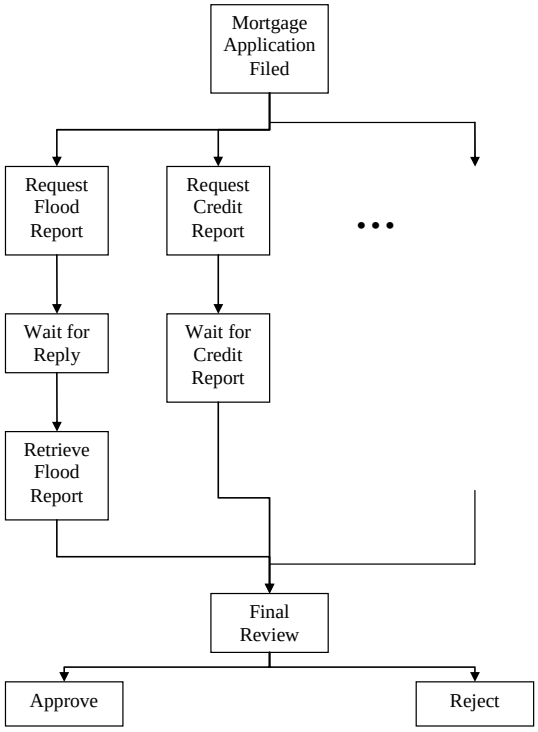

Figure 2. The business process at NationalMort incorporates content produced by business partners

NationalMort and its partner firms communicate using SOAP-based web services. CMIS defines the standard for content management interoperability.

Business process development environments (including the EMC® Documentum® BPM Suite) allow developers to build processes such as those depicted in this illustration using graphical display tools. When such steps as "Request Flood Report" or "Retrieve Flood Report" are configured, the BPM tool accesses the WSDL for these services, and enables the developer to "wire up" portions of the input and output messages (as defined in the WSDL) to the rest of the business process.

The two branches in Figure 2 demonstrate CMIS utilization both in a content pull and a content push pattern.

Requesting a flood report—content pull

Various land surveying firms in different geographies have the technical expertise and local knowledge to provide NationalMort with the required reports. However, each land surveying firm has its own way of managing content. Each has its own repository.

To do business with NationalMort, a land surveying firm only needs to implement two simple web services. First is one that allows the NationalMort to request a flood report. The second is one that allows the mortgage company to retrieve the flood report through a CMIS interface into the surveying firm's repository.

When NationalMort requests a flood report, the request step of business process issues a web service request to the endpoint supplied by the flood report firm and includes in the request message the property

address and other relevant loan-specific information. This is an asynchronous request that does not involve CMIS.

The NationalMort business process waits for notification that the land surveying firm has completed the flood report and filed it within its own content repository. Asynchronous messaging can be implemented either via standardized mechanisms or proprietary means; in either case this is outside the scope of CMIS. For the purposes of this example it is only important that when the surveying firm notifies NationalMort that the report is ready, included in the notification is the OID of the report that is stored in their repository.

Thus the "Retrieve Flood Report" step is executed and makes a sequence of two CMIS web service requests. The CMIS Object Service provides operations that allow both the actual content and content metadata to be retrieved.

- The *ObjectService.getProperties* operation is invoked, passing in the Object Identity (OID) that was provided by the firm, to obtain the document metadata such as the generation date of the report.
- The *ObjectService.getContentStream* operation is invoked, passing in the OID, to obtain the actual document.

With these two operations the content is pulled from the repository of the outside firm, into the repository of NationalMort and associated with the remainder of the business process.

Without CMIS, there are likely to be many implementations of the "Retrieve Flood Report" step, perhaps one for each land surveying firm that is a business partner. NationalMort would have a difficult (and costly) situation managing multiple implementations. Instead, with CMIS, the "Retrieve Flood Report" step can be implemented once and functions regardless of the land surveying firm providing the flood report.

Requesting a credit report—content push

Interoperability has bi-directional business benefits. While NationalMort works with multiple credit reporting agencies, each of these firms also deals with many mortgage companies. CMIS ensures that both report requestors and report providers can expedite their operations through a seamless information exchange, regardless of the content repositories they are using.

When NationalMort requests a credit report, it simply provides:

- The endpoint for the CMIS compliant ObjectService repository interface
- The OID of the folder into which the credit report is to be filed

Each credit reporting agency can rely on CMIS, to develop a single activity for delivering credit reports to any mortgage company that allows delivery via CMIS interfaces, including NationalMort. When the agency is ready to delivery the credit report, it invokes the *ObjectService.createDocument* web service and provides the OID for the folder together with the document contents and the metadata.

Consequently, the CMIS interface between credit reporting agencies and mortgage companies is not prescriptive of repository implementations. It is designed to work with existing systems.

My Favorite Items: An adaptable personal portal

FleetlyImports, also a hypothetical company, is a logistics and freight forwarding firm serving North American distributors that are importing goods from South Korea, China, and South East Asia. The company competes on the excellence of its customer service, and its ability to solve import problems before they arise. Account executives work closely both with customers in the United States and Canada to manage the flow of goods across the Pacific.

Not surprising, each account executive needs to keep up-to-date with relevant information about his or her North American customers, their Asian manufacturers, and conditions across the Pacific that might affect commerce and trade. Each executive needs to interact with many different information sources that are stored in a variety of repositories and accessed over the Internet.

A Rich Internet Application

FleetlyImports has thus developed a web application that includes a My Favorite Items (MFI) feature, an element of the user's personalized portal page that allows executives to interact seamlessly with content being managed in multiple repositories. The application has been developed using an Ajax style of programming, giving the user a Rich Internet Application experience—one that has the interactivity of a desktop application and the deployment advantages of a web-based application.

Account executives use a wizard to configure their MFI experience. When personalizing the portal, they navigate through the repositories to which they have access, and select both documents and folders that they frequently need to access. They can then display the content from the multiple sources within the MFI tile, together with a hyperlink to the content and relevant metadata about the items—such as date last modified, goods categories, and document/folder status. Furthermore, account executives can create, edit, and delete documents, managed within disparate repositories, directly from the MFI interface.

Because MFI is implemented with an Ajax style of coding, with a good portion of the application code running directly in the browser, REST-based bindings are the most suitable web services protocols for CMIS. The application framework of the browser does not offer SOAP or WS-* support but does provide the toolset needed for RESTful interactions—that is, HTTP, XHTML, DOM, and XSLT.

Mashup development typically involves a great deal of custom coding to account for the differences in data models and APIs of the data sources that are being brought together. CMIS addresses data model differences by defining the core document and folder object types, including a standard set of metadata for each type of object. CMIS further eases the programming burden by providing a single programming interface that works with any CMIS compliant repository, regardless of the implementation details (platform, API language, and so on) of the underlying repository. As a result, an application developer can add additional content repositories to the portal environment without modifying the portal code.

When developing the MFI application, the developer codes against the REST binding of the CMIS specification. The developer does not need to know the details of the underlying content repositories.

The MFI configuration wizard

When an end user runs the MFI configuration wizard, the CMIS Navigation and Object Service interfaces are used to provide the following sets of interactions:

a) The code displays a list of accessible repositories.

b) The end user selects one of these repositories. An HTTP GET request is sent to the URL of the Atom Service Document for that repository. The response includes a URL for the repository's root folder.

c) The name of the root folder is retrieved with an HTTP GET request to the repository root folder URL. This property is returned in an Atom <entry>; this is an invocation of the *getProperties* operation of the CMIS *object service*.

d) The list of items contained in the folder is obtained with an HTTP GET request to this folder URL with "?getChildren" appended; this is an invocation of the *getChildren* operation of the CMIS *object service*. This request returns an Atom <feed> containing <entry> elements, which in turn contain properties, including name and URL, for each of the items contained in the folder.

e) The XML documents returned in steps (c) and (d) are then transformed into a display for the end user.

f) If an end user selects one of the items displayed, the URL for that item is stored in the MFI tile within the browser.

g) If an end user expands a folder (one of the children of the previously expanded folders), the action goes to step d) above.

MFI content displays

MFI displays content from disparate repositories within the separate tiles. The code for the MFI display dispatches a GET HTTP request to the resource address (URI) for each item selected during the configuration process described above, obtaining the NAME, LAST_MODIFICATION_DATE, and other properties for each item; each request is an invocation of the *getProperties* operation of the CMIS *object service.* The properties for each item are returned in an Atom <entry>. The <entry> elements from each request are then aggregated and an XSLT is applied to generate the MFI tile display. [4]

It is also straight-forward for the developer of the MFI tile to expose content management operations to the user directly through the tile display. For example, if an end user wants to delete an object through the MFI, the application developer would ensure that a link to the object including "delete" directives was included in the display. After prompting the user to confirm intentions, the application would invoke that link resulting in the issue of a DELETE HTTP request to the resource URL; this action invokes the *deleteObject* operation of the CMIS *Object Service*.

The end result of the MFI tile is a seamless, interoperable environment where account executives at FleetlyImports can manage the content they need to do business with globally distributed customers and manufacturers.

The impact of CMIS

Standardizing core ECM functions

It is important to recognize the significance of CMIS. As we advance deeper into the information age, the challenge has grown from content capture and storage to content sharing and exploitation. There is an increasing need for standardizing core functions among ECM vendors to allow reusing content that is stored in vendor-specific repositories. Customers are demanding interoperability among disparate content repositories. Vendors find that standardization increases the value of customers' investments and simplifies purchase decisions. Customers, in turn, benefit from the increased flexibility and choice among their technology providers.

Indeed, as a rapidly growing sector of the overall enterprise application market, the ECM industry is entering a standards era. Standardization brings discipline to a fragmented industry. This is a sign of technical maturity and sophistication. Standards steer investments and innovation in a common direction so that customers can better benefit from one another vendors' implementations. By reducing or eliminating redundant efforts, standards lower the costs of technology innovation, application development, and tool building. In the end, standards for enterprise content management break down barriers that inhibit information sharing; make technology and solution less expensive and more available; and accelerate the growth of the entire ECM industry. As illustrated by the continued evolution of Documentum capabilities,

[4] Note that if several of the selected items come from the same repository the MFI display code may implement an optimization that utilizes the CMIS *query service* to retrieve properties for multiple objects at once.

EMC fully supports industry standardization initiatives that benefit customers and increase the value of their investments.

In the quest for standards for sharing enterprise content, the first step is to be able to overcome content stovepipes—to access the vast amount of unstructured business information that is stored within proprietary repositories. It is therefore imperative to provide a simple way to interoperate with existing repositories. To this end, EMC, IBM, and Microsoft have joined forces to draft the CMIS specification, including protocol bindings for SOAP-based and REST-based web services.

The next step of this initiative is to engage with other vendors and technical experts within the framework of an OASIS technical committee. As one of the submitters, EMC plans to work with OASIS participants through the technical standardization process, to fine tune the CMIS submission, and to shepherd it to becoming a widely adopted standard.

Conclusion

CMIS and EMC Documentum

EMC's efforts are based on the recognition that the Internet and the Web now provide mainstream technology for information delivery. An ECM platform, such as the EMC Documentum content management platform, needs to leverage this technology to its full capabilities. An ECM platform needs to do more than simply use the Internet as a wire between a client and a repository. Rather, an ECM platform needs to exploit web-based capabilities—such as dynamic and scalable connections, firewall tunneling, enterprise metadata management, and loose couplings—to enhance users' experiences, to simplify application development, and to support large-scale deployment. As a result, CMIS becomes a stimulus to guide the industry towards a flexible application development paradigm.

EMC has begun the process of leveraging web-based technologies. Currently, Documentum includes Documentum Foundation Services (DFS), which expose Documentum content management functionality as web services, together with a Java SDK and a WSDL service interface. In support of CMIS, EMC has built a reference implementation of CMIS SOAP binding on DFS. A reference implementation for the CMIS REST binding is underway. Once standardized, CMIS will enable DFS to reach out to customers who want to develop interoperable content applications in a more cost-effective fashion than possible with current technologies. For Documentum and EMC, CMIS represents an important direction for future development initiatives.