# Content Model Architecture

## Audience

If you are only interested in using a Fedora Repository to store your content (and associated metadata) and access the content exactly in the format you stored it, you do not need to read this section. Fedora will use the Content Model Architecture behind the scenes and you do not have to do anything different from earlier versions of Fedora. However, if you want to use Fedora's full capability the CMA is the gateway. We will try to lead you through steps to understanding the CMA and what it can potentially do for you.

## Introduction

A major goal of the Fedora architecture has been to provide a simple, flexible and evolvable approach to deliver the "essential characteristics" for enduring digital content. Whenever we work with digital content, it is with an established set of expectations for how an intellectual work may be expressed. With experience we develop "patterns of expression" that are the best compromise we can craft between the capabilities of our digital tools and the intellectual works we create in digital form. We store our digital content with the expectation that all the important characteristics of our intellectual works will be intact each and every time we return to access them, whether it has been a few minutes or many years.

We also want to communicate our intellectual works effectively to others. To do this, we have an expectation that the important aspects of our digital works are delivered accurately to users accessing them. Often we teach the same patterns of expression used to create our works to aid our users in comprehending them. And the same patterns may be used once again to enable collaboration for creating new or derived works.

Librarians, archivists, records managers, media producers and the myriad other author and publishers of intellectual works have long used content patterns calling them, for example: books, journals, articles, collections, installations. The term "content model" originated with the publishing community to describe the physical structure of the intellectual work. Users and collectors of intellectual works often add value by organizing works, annotating them, and producing ways to find them or information within them.

With the development of digital technology, publishers, users and collectors have applied these same patterns to this new media with some initial success. But like many advancements, digital technology enables ways to create and use content in ways that cannot be achieved with physical media. While it is beyond the scope of this document to discuss the many emerging aspects of digital content technology and its use, there are two key subjects that help in understanding the requirements of the architecture presented here.

First, reusable patterns such as content models can reduce the effort to create or capture, ingest, store, manage, preserve, transform, and access digital content. For example, content models can be used for content classification to facilitate discovery. Other uses include content validation usually at ingest or modification. Content models can be, for example, used as a template when content is created to generate user interfaces, drive workflows, describe content components, or to manage policy enforcement.

Second, digital content is not defined by its format or technology, and may also incorporate functions as a part of its nature. For example, when we access a digital picture we experience its resolution and color fidelity; the technology that delivers that experience is irrelevant. A spreadsheet or a video game is hardly the same thing if there is no software to make it function. Those features which must be present to provide an authentic experience of the digital content are called the "essential characteristics" and they can be captured in a content model to ensure durability of the experience as format and technology changes over time. Those same characteristics also facilitate sharing by describing the nature of the content and the ways it can be used.

Similar needs have been noted in the software engineering community for the development of complex computer systems. Often, organizational information outlasts the technology used to create and access it. Corporate mergers and breakups raise havoc with the integration of company information technology infrastructures. The same concepts that have been developed to satisfy agile IT infrastructures can help provide solutions for creating, accessing and preserving content. In this document, we introduce the fundamental concepts of Fedora's Content Model Architecture or CMA. The CMA builds upon the basic building blocks established by previous versions of the Fedora architecture, restructuring them to both simplify use while unlocking their potential.

We use the term "content model" to mean both:

1. Content structure as used by publishers and other traditional content-related professions
2. A computer model describing an information representation and processing architecture

By combining these very different views, CMA has the potential to provide a way to build an interoperable repository for integrated information access in our organizations and to provide durable access to our intellectual works.

As we introduce CMA concepts, we will discuss the rationale behind the design decisions. This is only the first generation of the CMA and, like the rest of Fedora, we expect it to evolve. An understanding of design decisions behind this "first-generation" CMA is a key element for community participation in future generations of CMA development. Most important is an understanding of three significant and interrelated developments in software engineering: (1) object-oriented programming, (2) design patterns, and (3) model-driven architectures. It is beyond the scope of this document to discuss any of these developments in detail but we will make reference in this document to aspects of them which inform the design of the CMA.

## Content Model Architecture Overview

The Content Model Architecture (CMA) describes an integrated structure for persisting and delivering the essential characteristics of digital objects in Fedora. In this section we will describe the key elements of the architecture, how they relate, and the manner in which they function. The original motivation for the CMA was to provide a looser binding for Disseminators, an element of the Fedora architecture used to stream a representation to a client. However, the CMA as described in this document has encompassed a far greater role in the Fedora architecture, in many ways forming the over-arching conceptual framework for future development of the Fedora Repository.

The Fedora application community developed a number of clever approaches to add content model-like capabilities to Fedora. The CMA formalizes some of the "best of breed" techniques gained from further research and from our application community. Two primary ways of thinking about content models have emerged and both are supported by the CMA. The first approach is focused on using complex single-object models and is commonly called "compound." The second approach is to use multi-object models and is commonly called "atomistic" or "linked."

Prior implementations of the Fedora Repository utilized a set of specialized digital objects as a functional and persistence framework. All of these objects conform to the same basic object model. Digital objects in CMA are conceptually similar in prior versions of Fedora though some important implementation details have changed. Fedora still implements a compound digital object design consisting of an XML encapsulation (now FOXML 1.1) and a set of bitstreams identified by the "Datastream" XML element. We can also assemble multi-object groups of related digital objects as before using semantic technologies.

In the CMA, the "content model" is defined as a formal model that describes the characteristics of one or more digital objects. A digital object may be said to conform to a content model. In the CMA, the concept of the content model is comprehensive, including all possible characteristics which are needed to enable persistence and delivery of the content. This can include structural, behavioral and semantic information. It can also include a description of the permitted, excluded, and required relationships to other digital objects or identifiable entities.

The content model is expressed in a modeling language and stored in one or more bitstreams like any other kind of content. It may be useful to think of the content model as one or more closely related documents that contain a machine processable model.

There will likely be more than one content modeling language; the CMA does not specify that there be only a single standardized one. One of the key aspects of Fedora is the expectation that over time all things will change and the same will be true of content modeling languages. There are many valid approaches to content modeling and we wish to enable innovation in this area. Guided by these observations, the CMA is designed to be a framework for developing and deploying content model-driven repository infrastructures. However, while it is a Fedora first principle to minimize required elements, there are a small set of features, described below, that the content modeling language must provide in order for Fedora to function.

Since we anticipate that a large number of digital objects will conform to the same content model, they may be treated as a class. While we need to be careful about analogies between "class" in CMA and "class" in object oriented programming languages or in semantic technologies, exploiting the notion of "class" is a major objective of CMA.

We must have a unique, unambiguous method to identify the class. For this purpose, in the CMA we have defined the "Content Model object," a digital object that both represents the notion of the class and can contain the content model. We use the identifier of the content model object (or CModel) as the class identifier. Following the rules of Fedora identifiers, the identifier of the CModel object can be encoded within a URI. We will describe the rationale for this decision in a later section but this approach provides two immediate benefits: (1) it provides a scheme which works within the Fedora architecture with minimal impact, and (2) it is compatible with the Web architecture, RDF and OWL. We can even build functionality using just the knowledge of the identifier without creating a content model. Having a uniform method for identifying a digital object's class maximizes interoperability.

The CMA does not require that digital objects explicitly conform to its architecture or explicitly declare any of its metadata elements beyond providing well-formed Fedora digital objects - unless you want to use the advanced features provided by the Fedora repository. The CMA uses a "descriptive" approach where the Fedora Repository will issue a run-time error for any operation it cannot perform. In most cases, you should still be able to disseminate bitstreams exactly as they are stored. CMA's dissemination approach is more consistent with the Web architecture, and provides a better balance between durable access to content and future innovations.

The minimum requirement to participate in the CMA is for a digital object to assert a relation to record its class' identity. Digital objects that do not explicitly identify their class are assumed to belong to a system-defined "Default Content Model" which has a repository-defined reserved identifier. In CMA, you should use a digital object (see CModel below) as a way to register a "class" in a repository federation.

The remaining functionality is enabled by creating the content model and storing it within Fedora digital objects like any other content. This permits applications or the repository itself to disseminate the content model, interpret it and use it to provide functionality for the set of objects it describes. Content designers are free to develop content models (or even content modeling languages). Content Models and Content Model objects are designed to be shared. Digital objects having the same content model automatically form an interoperable information community. If the whole Content Model object is shared, keeping the object's identifier the same across multiple repositories, the community can easily extend across multiple organizations.

There are two basic approaches to using content models. First, the content and content models can be disseminated to applications able to interpret them to deliver the essential characteristics of the content. Disseminating the content model to external services can be also be used when creating new digital objects for ingest, validation, transformation and replication.

Alternately, the Fedora Repository can be used as a content mediation layer which interprets the content model to disseminate the content correctly. To accomplish this, the Fedora Repository must have access to code compatible with the Content Model. The compatible code may be added to the Fedora Repository using its plug-in architecture in combination with service mechanisms. While all the tools needed to plug in your own Content Model mediation are not complete in Fedora 3.0, this feature of the CMA will permit serving several generations of digital objects in the same repository reducing the need to update objects whenever there are new releases of Fedora. This will increase the durability of collections and enable longer periods between migrations of Fedora digital objects to new formats.

While the CMA does not force you to use a specific content modeling language, Fedora 3.0 contains a reference implementation that enables the Fedora Repository to operate much as it did in prior versions. The following sections describe CMA in more detail and provide instructions on how to use the reference content modeling language so you can create your own CMA compatible objects immediately. Over time Fedora Commons will support the development of one or more content modeling languages as part of solution bundles that may be used by the community with minimum effort.

## Specializing Digital Objects

One of the basic elements of the Fedora architecture is the Fedora digital object. Every digital object stored in a Fedora repository is some variation of the same basic Fedora digital object model. In Fedora, digital objects containing data (Data object) utilize an XML element called the Datastream to describe the raw content (a bitstream or external content). In Fedora 2 and prior versions, digital objects containing data may also have contained Disseminators. The Disseminator is a metadata construct used by Fedora to describe how a client can access content within the digital object or remotely referenced by the digital object. If you only needed to access the raw content, default functionality was provided by the Fedora Repository which did not require that a Disseminator be explicitly added to the digital object. Unfortunately, the older design meant that the Disseminator was repeated in every Data object.

In Fedora 2 and prior versions, three specializations of the Fedora digital object were used: the Data object, the Behavior Definition (*BDef*) and the Behavior Mechanism (*BMech*). The BDef contained a description of the access interface which, in turn, was implemented by the BMech. In combination, the Fedora digital object model provides a uniquely flexible way to persist and access digital content. However, these features were not as simple to use as they could be. In particular, repeating the Disseminator in every Data object made changes to the access interface very difficult since every object had to be changed.

In the CMA, the Fedora digital object remains the model for all digital objects just as it was in prior versions of Fedora. However, for the CMA we define four specialized variations of the Fedora digital object (see Table 1).

| Object Type | Code | Description |
|---|---|---|
| Data | Data | A container for content |
| Service Definition | SDef | A container for the service definitions, an element of a content model |
| Service Deployment | SDep | A container for service deployment bindings |
| Content Model | CModel | A container for content models |

**Table 1 - Fundamental Fedora Object Types**

Each of these specialized digital objects will be described in much greater detail below. The BDef and BMech have long been recognized as having reserved functions in Fedora and we often called them "control" objects because they contain data used to control object-specific functionality. CMA adds a new control object, the CModel. However, some of the data needed for object-specific functionality was located in the Data object (often just labeled generically as the digital object) in prior Fedora versions. In particular, Disseminators were defined in the Data object. The CMA eliminates the Disseminator and redistributes control data in a more logical and reusable fashion among the "control" objects.

Care should be taken in equating the names of object types in the Fedora 3.0 from the prior (Release 2 and earlier) Fedora architecture because there are some similarities to roles played by these objects in both the old and new architecture. However, the CMA is a major redesign so care should be taken not to automatically equate the roles of the control objects in the old and new implementations. In pre-release reviews with community members, we discovered that there were concerns about the potential of confusion so we have used a new naming scheme for Fedora 3.0.

Also, care should be taken not to equate the digital object as a container and any data (or metadata) it contains. Since the deployment, functional , persistence and information views of the Fedora architecture tend to intermingle more than in most architectures, it is easy to accidently conflate characteristics of the architectural elements across views. We will try to make the distinctions clear where needed.

Table 2 lists the supported relationships between fundamental object types in the CMA.

| Origin | Target | Relation | Purpose |
|---|---|---|---|
| Data | CModel | hasModel | Identifies the class and, optionally, the object containing a model of the essential characteristics of the class |
| CModel | SDef | hasService | Identifies the object containing a model of the functional characteristics of class members |
| SDep | SDef | isDeploymentOf | Identifies the object containing a model of the functions being deployed |
| SDep | CModel | isContractorOf | Identifies the object containing a model of the information being deployed |

**Table 2 - Relationships Between the Fedora Object Types**

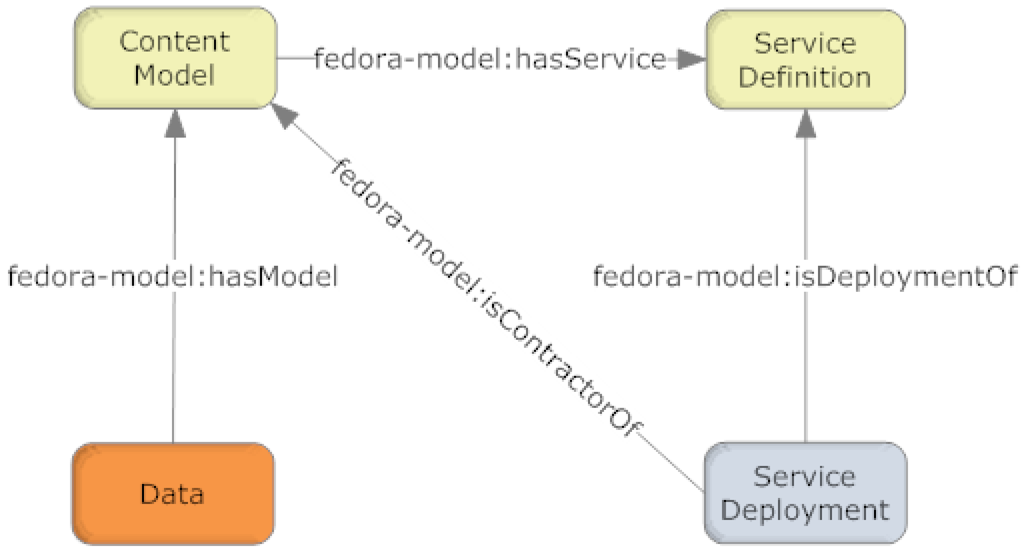

**Figure 1 - Fundamental CMA Relationships**

Figure 1 illustrates the required relationships between fundamental object types in the CMA. In the CMA object serialization these relations are asserted as RDF statements in the digital objects' RELS-EXT Datastream. These relations are asserted only in the object at the origin of each arrow though typically these relations will be harvested and indexed within utilities such as Semantic Triplestores or relational databases to enable fast query over them, or into caches which permit rapid access to their functionality.

The "hasModel" relation identifies the class of the Data object. There may or may not be a Fedora digital object that corresponds to the identifier. If the identifier refers to an object it must be a CModel object and contain the base content model document. It is expected that many Data objects conform to a single Content Model (and have a relation asserted to the same CModel object). The Content Model characterizes the Data objects that conform to it.

The SDef object describes a Service and the Operations it performs. Defining a Service is the means by which content developers provide customized functionality for their Data objects. A Service consists of one or more Operations, each of which is an endpoint that may be called to execute the Operation. This approach is similar to techniques found in both object-oriented programming and in Web services. The CModel object uses the "hasService" relation to assert that its' class members provides a Service (and its associated Operations). A CModel is free to assert relations to more than one Service. A Service may be related to many CModels.

Deployment of a Service in a repository is accomplished by using the "isDeploymentOf" relation to the SDef object. The Service Deployment (SDep) object is local to a Fedora repository and represents how a Service is implemented by the repository.

Finally, the SDep object asserts the "isContractorOf" to indicate the CModel (effectively the class of Data objects) for which it deploys Services. This permits the SDep to access the Datastreams in the Data object and user parameters when an Operation is called for a Data object.

**To see more information about the Fedora Digital Object Model, please go to:** Fedora Digital Object Model