

Documentum 5 Architecture: A Technical Overview

August 2003

© 2003 Documentum, Inc. All rights reserved.

Documentum, and the corporate logo are trademarks or registered trademarks of Documentum, Inc. in the United States and throughout the world. All other company and product names are used for identification purposes only and may be trademarks of their respective owners. Documentum cannot guarantee completion of any future products or product features mentioned in this document, and no reliance should be placed on their availability.

This document is Documentum Confidential and intended for registered customers of developer.documentum.com. It is not to be reproduced or dissemination by any means to non-customers of Documentum, Inc.

Printed in the U.S.A. Part Number 60450803V1

Table of Contents

Chapter 1	Enterprise Content Management	9
	Content Applications	10
	The Building Blocks of Enterprise Content Management	12
	Pervasive Content Management	12
	Managing the Content Lifecycle	13
	Creating and Capturing Content	14
	Managing Content	16
	Delivering Content	17
	Archiving Content	19
	Processing Content	19
	Creating Content Applications	21
	Completing the Digital Value Chain	21
Chapter 2	System Architecture	25
	The Four Layers of the Documentum ECM Platform	25
	Content Repository and Services Layer	26
	Documentum Content Repository	26
	Content Objects	29
	Content Server	30
	Repository Services	30
	Library Services	32
	Content Management Services	34
	XML Content Management Services	35
	Process Automation Services	36
	Extended Services	38
	Media Services	38
	Content Intelligence Services	40
	Content Exchange Services	42
	Site Delivery Services	44
	Interface Layer	47
	Documentum Foundation Classes	47
	Documentum Business Object Framework	48
	Web Services	49
	Standard-Based Interfaces	50
	Client Layer	50
	Microsoft Windows-Based Applications	50
	Documentum Web-Based Applications	52
	Content Services Integration with Enterprise Applications	53
Chapter 3	Developer Tools	55
	The Development Life Cycle	56
	Configuring Content Applications	57
	Documentum Administrator	58
	Documentum Application Builder	58
	Developing Content Applications	59

Table of Contents

	Documentum Foundation Classes.....	61
	Business Objects Framework	63
	Type-Based Business Objects	64
	Service-Based Business Objects.....	65
	Data Access APIs	66
	Documentum Web Development Kit.....	67
	Documentum Desktop Development Kit.....	69
	Documentum Portal Integration Kit.....	71
	Documentum Media Services SDK	72
	Deploying Content Applications	74
Chapter 4	Enterprise Platform Fundamentals	77
	Open	77
	Extensible	77
	Scalable	78
	Multitiered Deployment	78
	Horizontal and Vertical Scalability	79
	High Performance	81
	Minimizing Connections	81
	Minimizing Data Transfer.....	82
	Reliable	84
	Secure	85
	Portable	86
	Global	86
	Comprehensive.....	88
	About Documentum	89

List of Figures

Figure 1–1.	Content Applications Span Industries.....	10
Figure 1–2.	Content Applications Span the Enterprise	11
Figure 1–3.	Pervasive Content Management	13
Figure 1–4.	The Content Lifecycle.....	14
Figure 1–5.	Integration with Authoring Tools	15
Figure 1–6.	Version and Rendition Management	16
Figure 1–7.	Virtual Document Management	17
Figure 1–8.	XML Publishing	18
Figure 1–9.	Content Lifecycle Management	20
Figure 1–10.	The Digital Value Chain	22
Figure 2–1.	Basic Documentum Architecture.....	26
Figure 2–2.	Content Repository Structure	27
Figure 2–3.	Globally Distributed Content Repository	28
Figure 2–4.	Documentum Document Object	29
Figure 2–5.	Documentum Object Hierarchy and Sample Attributes	30
Figure 2–6.	User Authentication.....	32
Figure 2–7.	Version Tree	33
Figure 2–8.	Object Relationships.....	33
Figure 2–9.	XML Content Management	35
Figure 2–10.	Documentum Workflow	37
Figure 2–11.	Media Services Architecture	39
Figure 2–12.	Extracting Information From a Document With Content Intelligence Services	41
Figure 2–13.	Conceptual Classification	41
Figure 2–14.	Content Exchange Services.....	43
Figure 2–15.	Inter-Enterprise Workflow Services.....	44
Figure 2–16.	Publishing to a Web Site With Site Caching Services	45
Figure 2–17.	Site Deployment Services Builds on Site Caching Services	46
Figure 2–18.	Documentum Application Programming Interface (DAPI)	47
Figure 2–19.	Simple Client and Server Architecture	48
Figure 2–20.	Product Business Object.....	49
Figure 2–21.	Documentum Desktop	51
Figure 2–22.	Documentum Webtop	52
Figure 2–23.	Content Services for SAP	54
Figure 3–1.	Development Life Cycle for Content Applications.....	57
Figure 3–2.	Documentum Application Builder	59
Figure 3–3.	Applications Involve Coordinated Customizations at All Levels.....	60
Figure 3–4.	Documentum Development Tools	61

Table of Contents

Figure 3-5.	Documentum Foundation Classes	61
Figure 3-6.	Custom Object Types.....	64
Figure 3-7.	Type-Based Business Object	65
Figure 3-8.	Service-Based Business Object.....	66
Figure 3-9.	Documentum JDBC Services	67
Figure 3-10.	Documentum Web Development Kit (WDK) Architecture	68
Figure 3-11.	PIK Architecture.....	72
Figure 3-12.	Media Services Plug-Ins.....	73
Figure 3-13.	Deploying Content Applications as DocApps	75
Figure 4-1.	Three-Tier Deployment	79
Figure 4-2.	Connection Pooling	82
Figure 4-3.	Webtop Streamline View	83
Figure 4-4.	Content Server Clustering.....	85
Figure 4-5.	Web Content Language Fallback Rules	87
Figure 4-6.	Documentum Universe	89

List of Tables

Table 3-1.	DFC Packages	62
Table 4-1.	Supported Configurations	86

Enterprise Content Management

Business information exists in many forms: text documents, spreadsheets, images, XML files, Web pages, streaming video, streaming audio, e-mail messages, instant messages, and fixed content such as reports, records, and scanned images. From engineering drawings and manufacturing procedures to marketing collateral and sales presentations, unstructured content is critical to the smooth and efficient functioning of a company.

An enterprise content management system provides order to unstructured information. It manages the creation, management, processing, delivery, and archival of any content according to user-defined business rules. It establishes relationships between pieces of content, allowing the same content to be used in different contexts and renditions. It adds intelligence, creating categorization schema and metadata that make search and retrieval faster and more efficient. It automates the processing of content through its life cycle. It facilitates publication of content through multiple channels; for example, the same content can be published to a Web site, broadcasted as a fax, printed as a text document, and sent to a handheld wireless device. It promotes integration between departments and systems that previously worked within silos.

Documentum 5 is a robust, flexible platform that supports enterprise content management applications. Documentum 5 is a set of products and services that work together, in varying combinations, to meet the content management needs of an enterprise. The Documentum platform makes it easy to customize applications to meet specific business needs or to build custom content applications.

A platform provides development and runtime services supporting the common needs of a variety of applications. A platform provides a high-level interface to key functionality, so that the developer can focus on solving the business problem. For example, a database platform provides the services common to applications needing structured data storage, such as creating and maintaining data structures, controlling concurrent access, and returning query results. A Web application server platform provides an infrastructure for deploying Web-based applications, handling such essential tasks as managing memory and sessions, controlling user security, and providing an administrative interface.

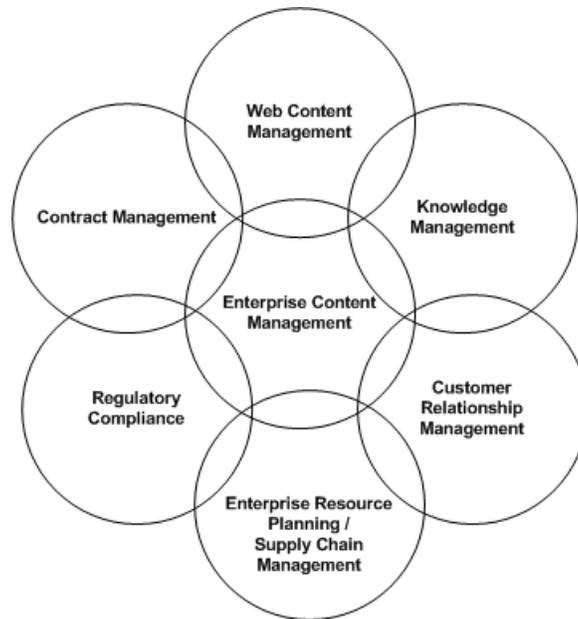
The Documentum 5 platform offers development and runtime services specifically supporting the needs of content applications. The architecture ensures that these capabilities work in an orchestrated fashion, enabling an enterprise to integrate its applications into a complete enterprise value chain.

This document provides an overview of the Documentum platform. It identifies the core capabilities and features required in a content management system and describes how the Documentum system architecture supports these capabilities. It also describes the development environments available for customizing and building Documentum-based content applications.

Content Applications

Content applications are any applications that use unstructured content, such as documents, images, e-mail messages, and Web pages. The range of applications that use some form of content services is extremely broad, comparable to the range of applications that use a database platform for managing structured data.

Figure 1-1. Content Applications Span Industries



Examples of content applications are:

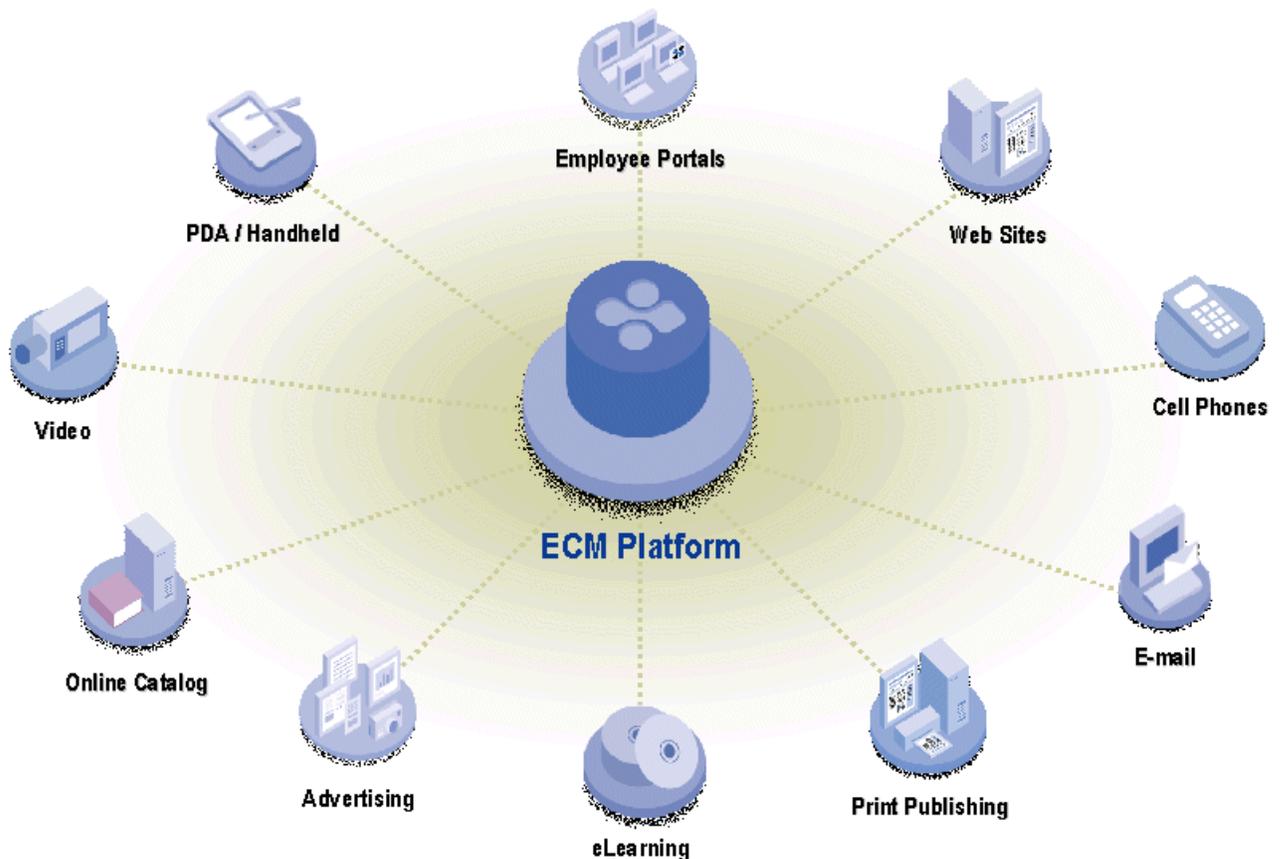
- **Contract management** applications track contracts through multiple versions, enable multiple people to collaborate on writing and reviewing them, enforce a strict approval process, and allow boilerplate text to be shared between documents.
- **Web content management** applications track contributions to a site from multiple sources, manage updates, and handle transformations that render XML into HTML using XSL stylesheets or create HTML renditions of documents authored in other formats.
- **Knowledge management** applications provide a single point of access for information stored in a multitude of formats and repositories.
- **Regulatory compliance** applications enable companies in regulated industries such as pharmaceuticals and financial services to meet mandated standards for project-related documentation and associated business records.
- **Enterprise Resource Planning (ERP) and Supply Chain Management (SCM)** applications automate business processes that often involve unstructured content as well as structured content. For example, the specification for a manufactured product can include a CAD drawing in addition to a bill of materials, and personnel records can include performance review documents as well as job and salary information.
- **Customer Relationship Management (CRM)** applications manage customer data such as contact information and records of customer interactions made during sales

or support calls. Much of the valuable customer information is unstructured, such as e-mail messages, faxes, or service order forms.

The first four applications are examples of *content-rich* applications, for which the management of content is a central concern. The last two are *content-enabled* applications, for which unstructured content enhances the structured data they manage.

Most companies use many of these application types — and others such as correspondence tracking or technical documentation management and publishing — somewhere in their organization. These applications may be packaged solutions or may be custom applications developed specifically for the company. Documentum 5 is the only enterprise content management platform that supports the complete range of content applications. It enables companies to integrate their various content applications across the enterprise, thereby increasing productivity and lowering total cost of ownership by simplifying system management. The integration can even extend to suppliers, distributors, and customers.

Figure 1-2. Content Applications Span the Enterprise



The Building Blocks of Enterprise Content Management

Companies succeed based on how well they manage their information, getting it to the right people at the right time. Everyone benefits from increased connectedness. The Documentum enterprise content management platform provides a foundation for this vision of a connected enterprise with the four key building blocks of enterprise content management:

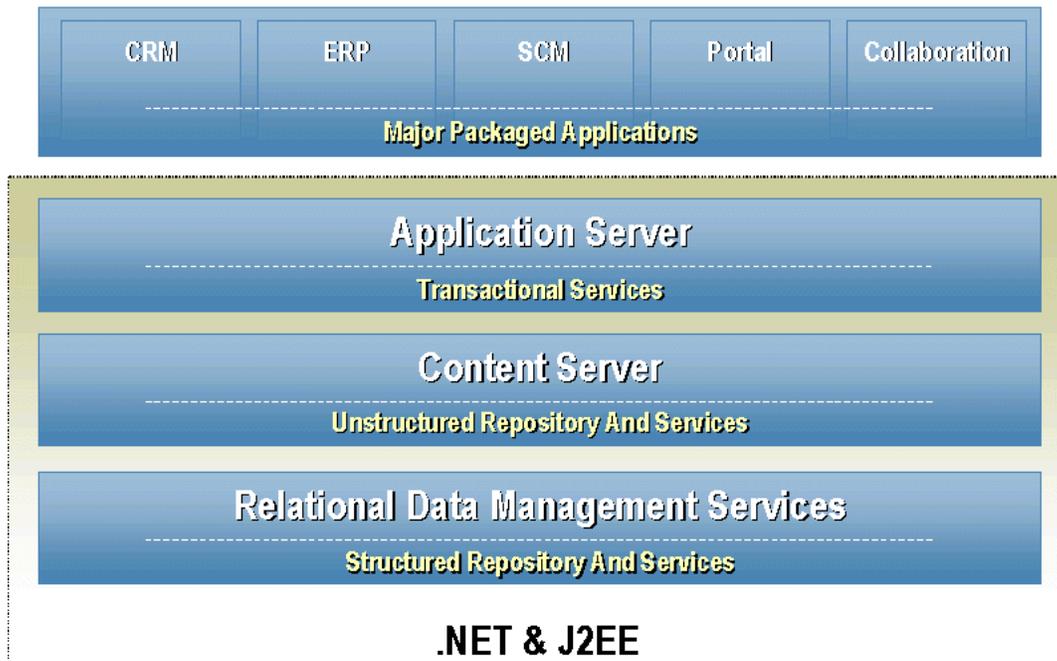
- Pervasive content management
- Complete content lifecycle
- Creation of content management applications
- Connected content management applications, completing the digital value chain

Pervasive Content Management

Pervasive content management is the ability to manage all content types anytime, anywhere. Documentum 5 can store content files in all known formats, including rich media formats, and is easily extensible to new formats. Documentum has the ability to capture and natively manage virtually any type of knowledge — documents, Web content, XML, rich media, fixed content (such as reports and records), collaborative content (including instant messages, discussion threads, e-mail, and more). Documentum integrates out of the box with many of the popular content authoring tools used today to capture content. The Documentum platform handles all phases of managing content, from creation, to management, to delivery, archival, and disposal, as regulated by laws and corporate policy.

Pervasive content management also means working effectively with other components of the infrastructure, such as the operating systems, programming tools, relational database management systems, Web application servers, authentication services, and enterprise applications such as ERP and CRM. Documentum connects seamlessly to these systems and provides a distributed repository so that companies can access and deploy their content any time, any place, around the world.

Figure 1-3. Pervasive Content Management



Managing the Content Lifecycle

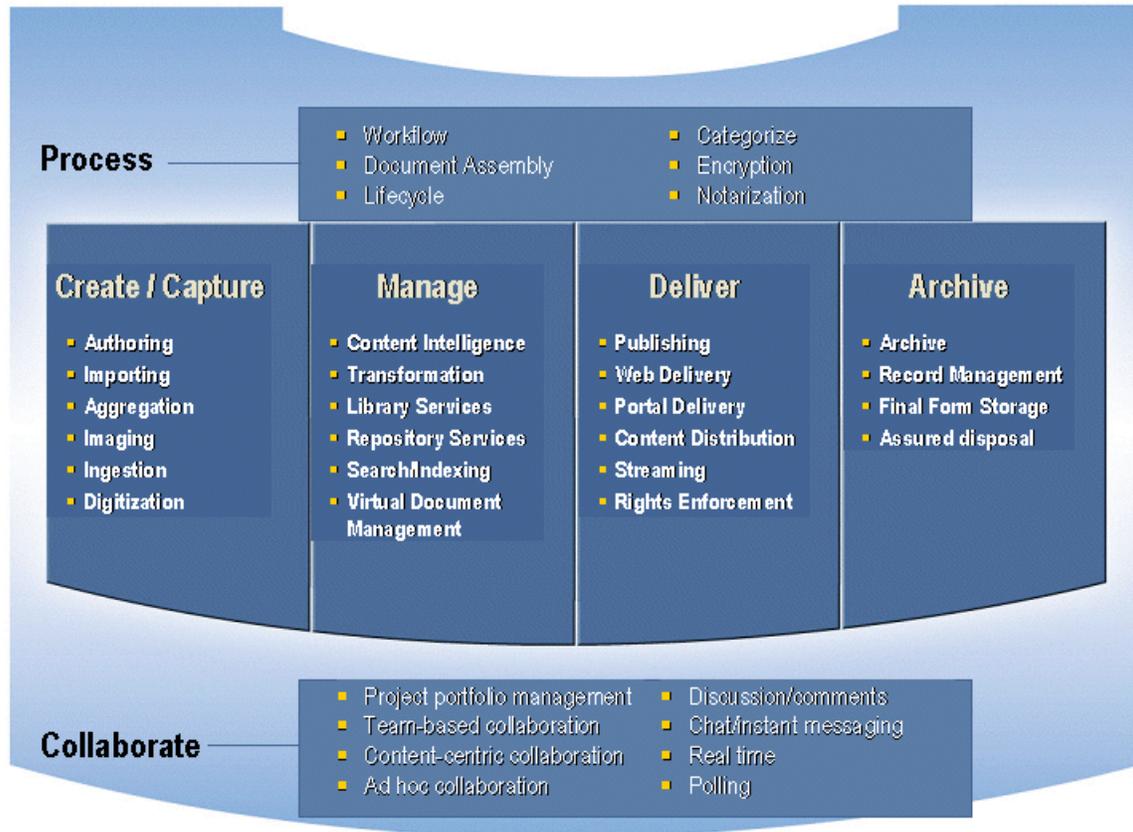
The Documentum platform can manage content from the moment it is created or captured all the way through to its ultimate destination. The objective might be publishing product information to a corporate Web site, distributing engineering specifications to subcontractors, circulating new sales incentives to field representatives via wireless devices, or delivering invoices to customers. Ultimately, the end point might be the authorized archival or disposal of content that is beyond its useful life.

How content is created and managed is just as critical as where it is published. While content applications differ in the types of content they use, they share a common lifecycle for the content. The lifecycle has four major stages:

1. Creating and capturing content
2. Managing content
3. Delivering content
4. Archiving content

Documentum 5 has core capabilities supporting each of these lifecycle stages, including tools for collaborating on content and automating business process workflows.

Figure 1-4. The Content Lifecycle

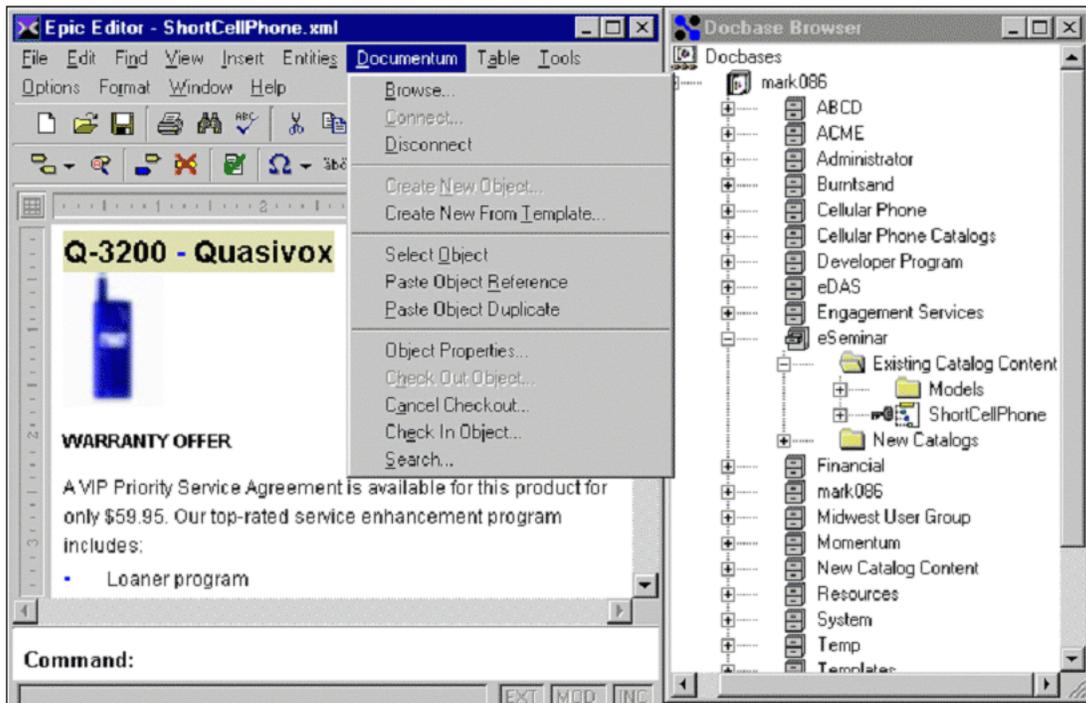


Creating and Capturing Content

The first job of a content management system is to collect the relevant content and add it to the corporate content repository. Content comes from a variety of sources, both internal and external to the company.

Documentum 5 integrates with authoring applications such as Microsoft Office products, Adobe publishing products, XML authoring tools, and CAD applications, enabling application users to add and retrieve content files directly from content repository. The integration uses standard protocols and interfaces such as WebDAV and ODMA. Documentum Web Publisher provides a powerful tool for creating Web content as well as managing it.

Figure 1-5. Integration with Authoring Tools



Documentum 5 also provides tools for aggregating and importing large volumes of content from disparate sources, including ERP/CRM systems, e-mail systems (such as Microsoft Exchange or Lotus Notes), and other enterprise applications. It supports document scanning, on an ad-hoc basis or production scale, to convert critical paper-based information into electronic content that can be managed as part of the content management system.

Documentum provides an environment in which users can communicate, coordinate, and collaborate as they develop content. Users can use virtual whiteboards, threaded discussions, and instant messaging to work together, simultaneously taking advantage of content management features to store and control the content they develop. Documentum eRoom Enterprise delivers a universally accessible Web-based collaborative environment that exposes content management services. Integrating collaboration with content management allows distributed teams to more effectively plan, strategize, make decisions, and build consensus as they design new products, coordinate their supply chain, engage clients, and work on other key business initiatives. eRoom is designed to capture and preserve project content as the project grows in scale and scope. Users can easily save content to a Documentum repository and create a link within the eRoom work environment, ensuring that content is securely stored while providing controlled access to authorized users. Storing content in the repository also enables users to apply content services such as workflow and lifecycle management or automatically publish approved content.

Managing Content

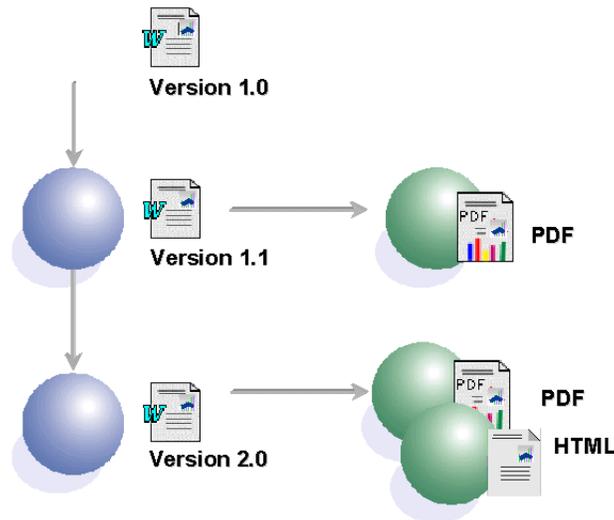
The content repository is the foundation of the Documentum content management system. The content repository is a secure storage area that provides organized access to the content, regardless of the source of the content or its format.

Documentum 5 can store content files natively in all known formats, including rich media or compound formats, and is easily extensible to new formats. The repository tracks an extensive set of attributes or properties about each content item. These attributes serve as metadata describing the content. The repository uses the metadata to organize the content, and users can use it to search for content that is relevant to them. The set of attributes stored for each item is configurable and fully extensible.

Each item in the repository is protected by powerful and flexible security that control who can access the content and what level of access each person has. Documentum 5 can control access with user- or role-based security. Content can be encrypted in the repository or when it is delivered to a user. Documentum 5 can secure content beyond the repository with solutions (such as digital rights management and records management, LDAP, SSL, and digital certificate support), essential for electronic submissions and secure e-commerce.

Documentum 5 provides automatic versioning capabilities to control, manage, and track multiple versions of the same content. It provides check-in and check-out capabilities that protect documents during editing to prevent conflicting edits. It tracks major and minor document versions. It can create renditions in multiple formats, such as PDF and HTML, for delivery through any channel or device, and to automatically update these renditions when the original document is modified.

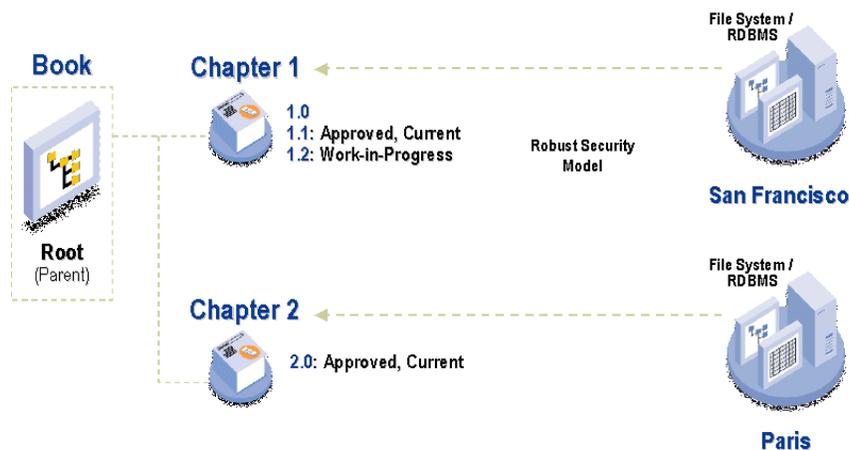
Figure 1-6. Version and Rendition Management



Documentum can manage links between related content and treat content in multiple formats as part of a single document, called a compound document or virtual document. For example, a Web page might consist of HTML text, images, and a stylesheet; in turn, the Web page might be part of a larger unit such as a product catalog. A Microsoft Office

document might include links to other Office documents, such as an Excel spreadsheet embedded within a PowerPoint presentation. Virtual document management quickly assembles information from across the enterprise into custom documents. For example, it enables the quick assembly of electronic common technical documents (eCTDs), essential to e-submission of new drug applications (NDAs). It automatically handles existing links — when the PowerPoint document described above is added to the repository, the linked Excel file is also stored. Any or all of a virtual document's contained documents can be assembled for publishing or perusal. Assembly and publishing services can be integrated with popular commercial word processors and publishing tools. The assembly can be dynamically controlled by business rules.

Figure 1-7. Virtual Document Management



Documentum provides the ability to automatically parse, validate, transform and map incoming XML documents.

The Documentum content repository supports the scaling and administration tools necessary for enterprise-wide data storage, including distributed physical repositories, load balancing, backup and recovery, and auditing.

Delivering Content

Content delivery can take two forms, sometimes referred to as “pull” and “push.” Documentum provides the means for users and content applications to access the repository and “pull” content from it. It also provides a variety of publishing options that “push” content to other forums from which users can access them.

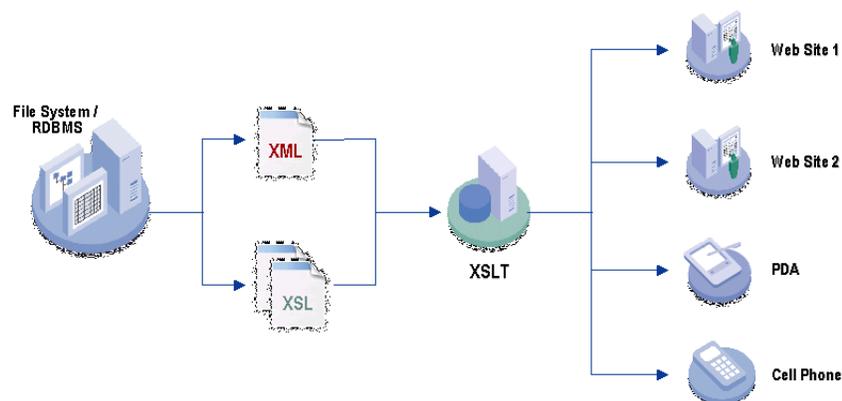
Content applications have a heterogeneous population of users — from IT personnel and developers to non-technical staff in many functional departments. Documentum 5 provides appropriate interfaces for each group: power users need a full-featured interface to the content repository; content contributors need an intuitive interface that makes sense in the context of the business applications they use; content consumers need a straightforward way to search for relevant information in large repositories without knowing how that information is organized or stored. Documentum products can make content available through specialized client applications, portals, and integrations

with enterprise applications. Documentum also offers products for specialized content applications, such as digital asset management, Web publishing, or document control for compliance.

Content publication takes many forms in today's enterprises. In addition to traditional printed documents and electronic document sharing, vital content is delivered electronically in a variety of forums, such as corporate Web sites, enterprise portals, and business-to-business or business-to-consumer distribution. Documentum enables a company to publish content through all of these channels, using the same source content. For example:

- Documentum Site Deployment Services (SDS) retrieves a Web site from a content repository and deploys the site to multiple servers or Internet service providers.
- Documentum Content Distribution Services provides a framework for distributing content to people both inside and outside of the company firewall. People *subscribe* to the content based on offers made available by the content provider. This product can automatically inform interested parties about updates to a repository and forward them the new or updated documents.
- Documentum Web Publisher enables non-technical users to create, manage, and publish content to one or more multilingual Web sites. Users create Web content in one of the many popular authoring tools with which Web Publisher integrates or they can create Web content with the Web Publisher native XML editor. Web Publisher uses workflows and lifecycles to manage Web content. During specified states in a Web content lifecycle, Web Publisher transforms content to HTML, merges the content with predefined Web page templates, and publishes Web pages to a Web server.
- Documentum provides the full range of publication services for XML documents, enabling companies to take full advantage of the power of rule-based XML processing. It can automatically divide XML documents into individual entities based on customer-defined rules. With its complete XSLT support, Documentum can publish single-sourced XML documents into multiple formats.

Figure 1-8. XML Publishing



Archiving Content

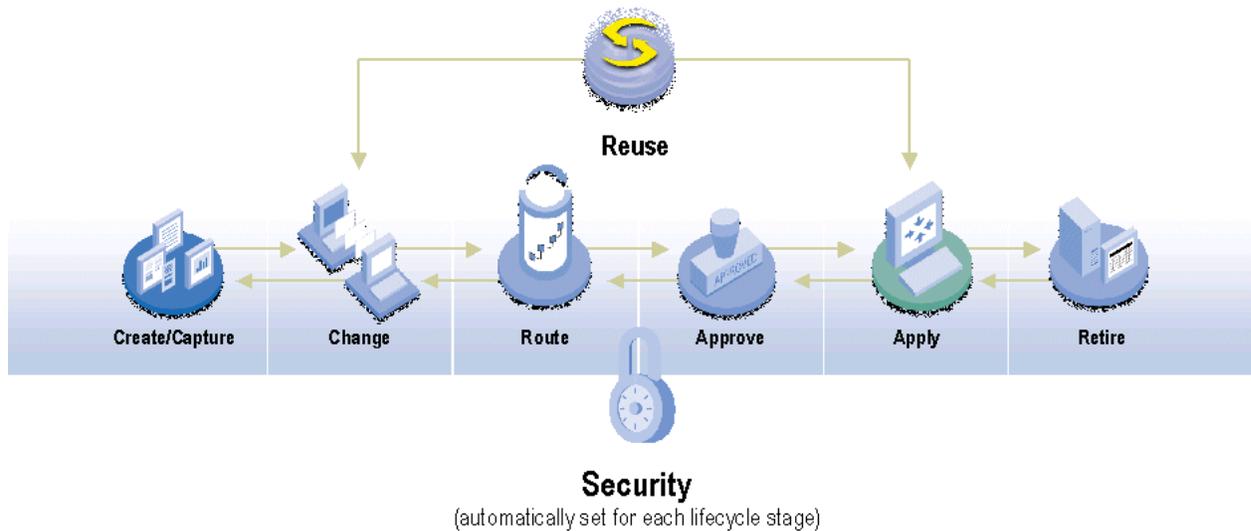
Companies today need to be able to preserve content in a trusted, scalable, and cost-effective way. Documentum provides a flexible architecture that enables integration with any archival and storage system. Since the Documentum repository relies on a customer's underlying operating system and database, Documentum transparently supports any storage system exposed through a file system interface and supported by any of the relational databases that the Documentum repository supports. As a result, Documentum customers can take advantage of any type of storage infrastructure they choose, including JBOD, RAID, CD and DVD jukeboxes, optical laser disks, and tape data storage as well as sophisticated networked storage systems such as network attached storage (NAS) or storage area networks (SAN). Additionally, Documentum can natively support storage systems exposed through a proprietary API such as EMC Centera, a content addressed storage with built-in immutability and non-repudiation.

With companies required to meet a growing body of regulations governing electronic information, the content in the repository must be classified and stored. A company may be required to produce records on demand, recover deleted content, or prove that missing records and content were disposed of in accordance with law and corporate policies. Documentum's record management features enable organizations to cost-effectively archive or dispose of records after their administrative, regulatory, or legal justifications have elapsed. Organizations can implement rules-based policies, set event-based as well as absolute retention periods, and implement "holds" for suspending the records review and destruction cycle.

Processing Content

Many content assets within an enterprise follow a consistent path through the content lifecycle: content is created, reviewed, revised, and approved, then used and ultimately superseded or discarded. Documentum can automate the stages in the content's life and the business processes for each stage. Workflows formalize the steps in a business process; lifecycles define the business rules for changes that apply to content as it moves through the stages of its life (such as Draft, In Review, Active, and Obsolete). Documentum can define and automate business processes associated with creating and distributing documents, including the ability to facilitate collaboration with outside partners and suppliers. For example, organizations can automate document workflow and lifecycle processes to enable compliance with records management policies and ISO certification procedures. Applications can require users to electronically sign off a document before passing the document to the next activity in a workflow or before moving the document forward in its lifecycle.

Figure 1-9. Content Lifecycle Management



Documentum's Inter-Enterprise Workflow Services extend automated workflow processes to external participants, enabling companies to integrate partners into cross-enterprise collaborations. Partners can exchange content over the Internet, completely independent of their proprietary systems. They participate in workflows and lifecycles when they receive and act on e-mail messages or when e-mail triggers a workflow automatically at each partner's organization. Inter-Enterprise Workflow Services allow the company to maintain complete control of a workflow even when associated tasks are performed by outside suppliers or partners. Additionally, Documentum workflows can integrate with other workflow systems, including enterprise business process management and EAI solutions such as BEA WebLogic Integration or TIBCO BusinessWorks.

One of the most difficult aspects of maintaining a large knowledge repository is organizing the information in a way that makes it easy for users to find. Users can locate documents based on their location in the repository or on the values of its attributes. If a document is in the wrong place or has incorrect attribute values, users may never find it. Documentum provides automated content analysis, classification, and categorization, which help extract information from unstructured documents and make it available for users. It can perform a semantic analysis that determines what each document is about, resulting in a list of the concepts discussed in that document. It can use the results of the analysis to automatically set values of document attributes or link the documents into appropriate locations.

Creating Content Applications

Documentum provides a complete, easy-to-use, consolidated development kit for any type of application requiring content management capabilities. It offers many options for integrating content management operations with existing client applications or rapidly building new applications, whether they are C, C++, Visual Basic, Java, Web server-based, or Web clients.

Documentum delivers pre-built content applications that can be *configured* or *customized*. Many aspects of Documentum content applications are controlled by text-based or XML-based configuration files, enabling customers to make significant changes to system appearance and behavior without writing any programming code. For example, customers can edit the text files containing the strings that appear in the user interface in order to “brand” the application in line with their corporate identity. For more extensive changes, customers can use Documentum developer tools to customize applications; for example, they can develop new controls, components, or Java classes. Components of the pre-built applications can also be integrated into new applications.

The Documentum repository includes a robust data dictionary, which is a collection of information about the repository and the objects contained within it. Applications can use the data dictionary to enforce business rules or provide assistance to users. For example, the data dictionary could specify that the value for a particular attribute must be unique, and applications can use that constraint when validating the data a user enters. By using the data dictionary, developers can ensure that all of the enterprise’s content applications use a uniform set of data validation criteria. The data dictionary also enables the grouping of attributes into categories that automatically control how the attributes are organized in the user interface.

Business logic encapsulated in core components can be exposed as Web services to provide access through other applications. Web services are a natural mechanism for integrating content management across systems, such as CRM, ERP, and portals.

Completing the Digital Value Chain

Documentum is a standards-based solution, making it straightforward to integrate applications with the rest of the computing infrastructure. Documentum is the only enterprise content management platform that provides the features necessary to support the full range of content applications, enabling companies to integrate various functions into a complete value chain. The digital value chain is the integrated set of applications and processes — within the enterprise and extending to suppliers, distributors, and customers — that support the creation, management, processing, distribution, and archival of any kind of electronic content.

Figure 1-10. The Digital Value Chain



The Documentum platform enables customers to integrate all types of content with packaged line of business (LOB) applications such as ERP and CRM applications. LOB applications are typically designed for handling a specific type of structured data for a specific purpose — for example, customer or inventory data. They usually have minimal capabilities for managing unstructured content, which can be limiting for users who often need to access, manipulate, or edit related unstructured content, such as invoices, contracts, SOPs, or safety data sheets. Integrating Documentum with these types of applications unifies the unstructured content across the enterprise with the structured content these applications manage, enriching the solution they provide.

For example, content-enabling a CRM application can connect a customer's purchasing history (structured content) with the contracts and correspondence (unstructured content) related to this customer. Because a customer service representative can instantly access any and all information relevant to the customer, the representative can provide more complete service, eliminate delays associated with searching for content, and offer other purchasing opportunities.

By content-enabling packaged applications, customers can begin to realize some of the most important benefits of enterprise content management, including increased efficiency and reduced costs. Everyone across the enterprise gains the ability to access and interact with all types of information in any format, regardless of which application created that content. That makes users more efficient and effective. And because content can be used and reused many times over for new and different purposes, the useful life of content is extended beyond its original purpose. That increases the value of content while reducing the cost to create it.

These benefits signal an evolution in content management from a departmental application to an enterprise deployment as enterprise content management becomes a distinct and essential layer in the technology stack, similar to the relational database layer (RDBMS) or the application server layer. The content management layer will ultimately become as important as the RDBMS layer because of the sheer volume of unstructured content that organizations are managing and the value of unstructured content to the entire enterprise.

Documentum backs up our commitment to this objective by investing significant resources in development and integration tools. The tools enable organizations to incorporate complete content lifecycle services in their corporate IT infrastructure stack. Developers can access these services within their preferred development environment and create content-based applications using the best programming language. And they can leverage the pre-built connectors between Documentum products and the other applications in the enterprise software stack, including portals, application servers, and packaged applications. Implementing the enterprise content management layer brings together all parts of the enterprise and, just as important, different organizations outside the enterprise — enabling everyone to work together more efficiently. That's when Documentum begins to truly unite the world through content.

System Architecture

“Architecture cannot be an afterthought.” — Howard Shao, EVP and Chief Technology Officer, Founder of Documentum

This chapter provides an overview of the key components of the Documentum architecture.

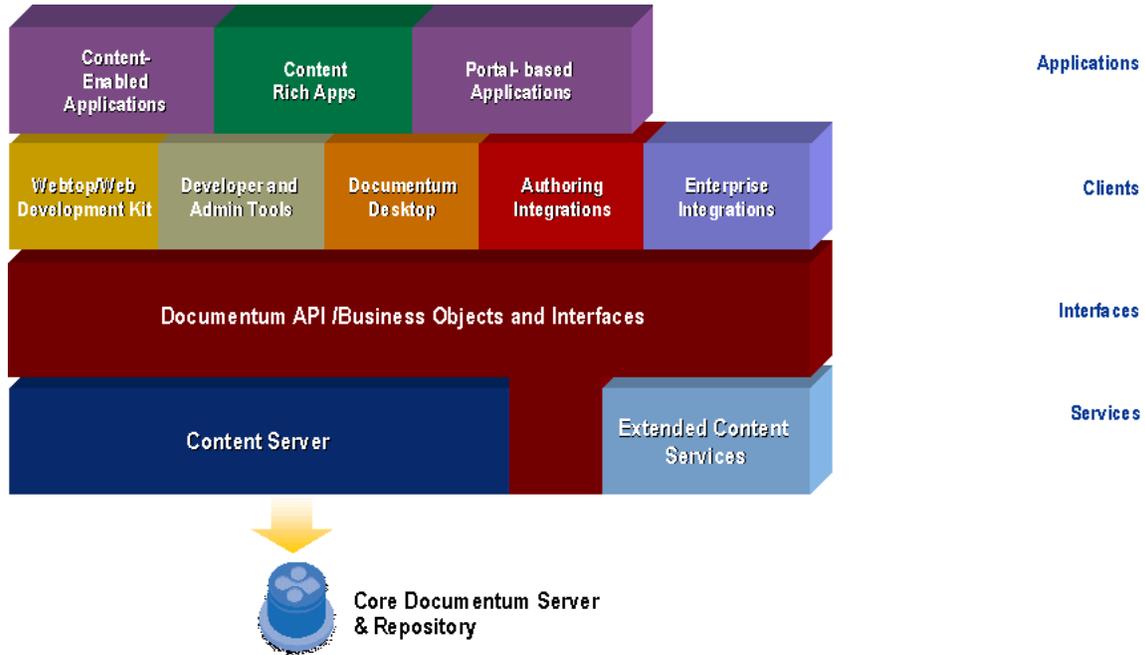
The Four Layers of the Documentum ECM Platform

Conceptually, the Documentum platform consists of four major layers:

- The services layer, consisting of Documentum Content Server and a variety of extended services, provides the content management functionality and serves as the foundation for all other products
- The interface layer, consisting of Documentum Foundation Classes (DFC) and its related APIs, provides communication between the services layer and the clients that use the services
- The client layer, consisting of end-user products, developer tools, and integrations with other systems, provides access to the content management functionality in the context of particular business functions
- The application layer, consisting of products from Documentum, its partners, or custom-built, provides the integrated applications that use content management functionality as part of their business solutions

The first three layers comprise the Documentum platform, supporting the application layer. Documentum also offers products that combine features from different layers to provide vertical solutions.

Figure 2-1. Basic Documentum Architecture



Content Repository and Services Layer

At the base of the Documentum platform is the enterprise content repository and Documentum Content Server, which manages the repository and implements the core content management capabilities. Content Server makes these capabilities available to clients and applications through the interface layer. Additional content management services can be added by installing various extended content services offerings, such as Content Transformation Services, Content Intelligence Services, Content Exchange Services, and Site Delivery Services.

Documentum Content Repository

Documentum Content Server is the software that manages the content repository and provides the fundamental content management capabilities. The Documentum content repository uses an extensible object model to store content and its associated metadata.

The enterprise content repository that Documentum manages is an abstract repository consisting of data stored in distinct physical sources; Content Server coordinates the different forms of data to create the object-based repository. Documents are composed of content files (the source file in its native format) and document attributes (also known as *metadata* or *properties*), such as document owner, version, and creation date. These attributes serve as metadata describing the content and the relationships between this content and other objects in the repository. The repository uses the metadata to organize the content, and users can use it to search for content that is relevant to them.

A Documentum repository can store its content files in any of these types of storage:

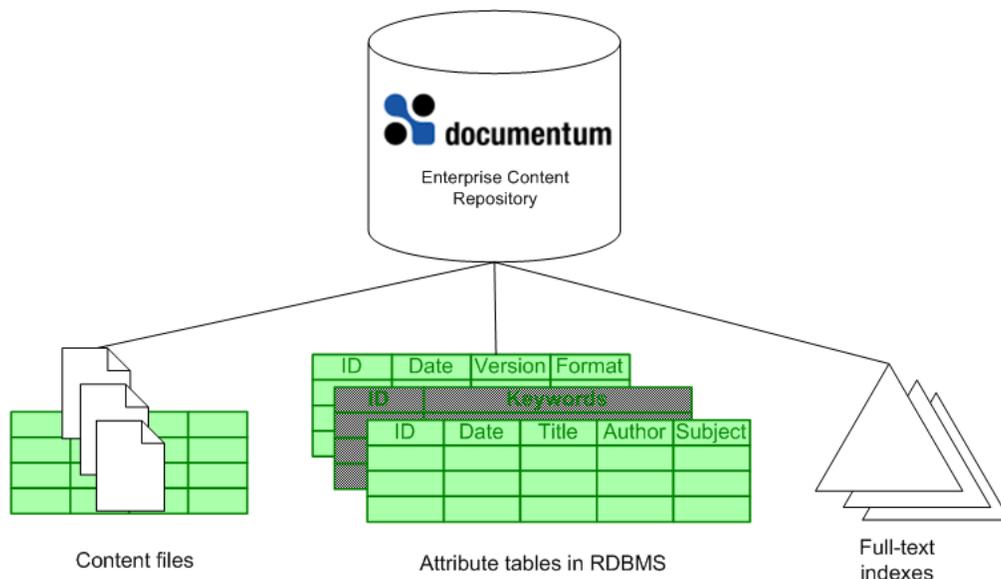
- In a directory structure in the server host's file system or an external storage facility. Content Server maintains its content store in a protected set of directories, so that users do not have uncontrolled access to them from outside of the Documentum system. Where security is paramount, the content can be encrypted using Trusted Content Services
- In a relational database management system (RDBMS) as Binary Large Objects (BLOBs) or data in varchar fields
- On a content-addressed storage device provided by vendors such as EMC or NetApp
- In an external storage area, such as a legacy system where content is outside the direct control of Documentum

Users and developers access any of these storage areas in the same way, so the underlying storage system is transparent. In fact, a given piece of content can move from one storage area to another as part of its lifecycle without requiring any changes at all to programming code or user navigation.

Document attributes are stored in tables in a relational database. The set of attributes stored for each item is configurable and fully extensible. It can include attributes required to have a single value, such as the document's globally unique identifier, and attributes that can have multiple values, such as keywords describing the content.

In addition to the content files and the attributes describing them, the repository includes a set of full-text indexes created by Content Server's embedded Verity full-text search engine. The full-text indexes enables content-based searching of the repository. When a document is added to the repository, the associated content files are added to the storage area's index the next time the index is updated. Indexing a document enables users to search not only the data from the document's content file but also selected attributes.

Figure 2-2. Content Repository Structure



Content Server integrates the content files and associated metadata into document objects and provides object-based access to the resulting documents. From the point of

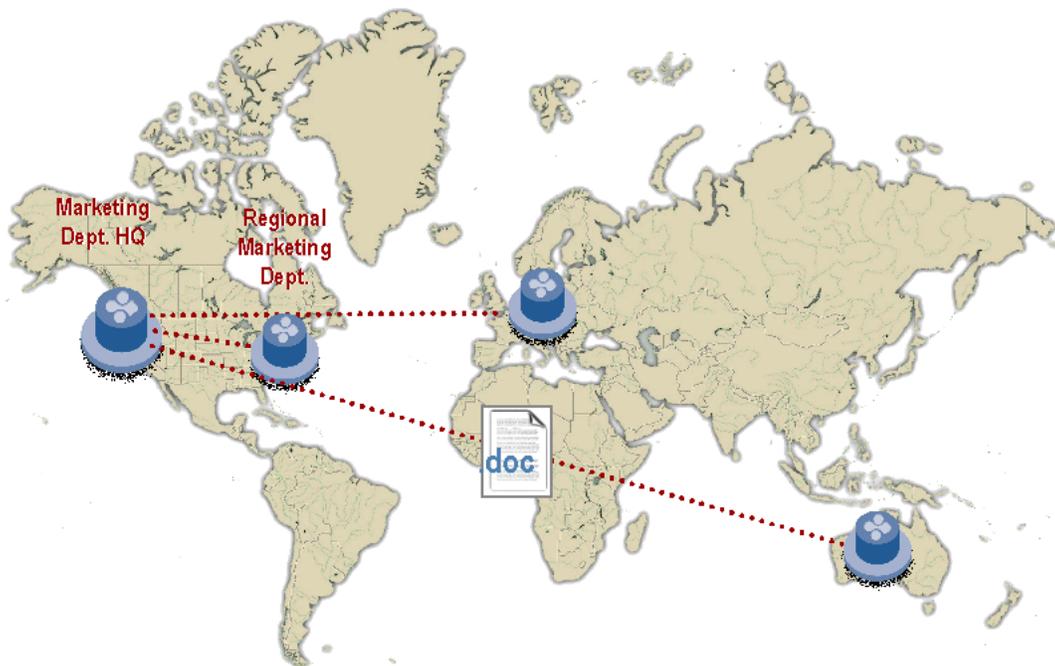
view of a client application, there is a single repository whose implementation details are irrelevant. The server treats the content files and metadata as part of a single entity and handles updates to the document object as a single transaction: it updates both elements in concert or updates neither of them. The server automatically updates the index entries as well, ensuring that the three types of data cannot get out of synchronization.

Content Server enforces proper security measures to ensure that only authorized users can access the content files, metadata, or indexes. No user or application can bypass the server to access the data directly through the file system or database software.

Content Server can also store information that is not directly related to particular content. Developers can register database tables with Content Server, thereby making them available for querying and updating through Content Server facilities.

Large enterprises can require a distributed repository or multiple Documentum content repositories. For example, a global company might have a Documentum content repository in each geographical region, with the goal of storing content locally to the users who work on it. Content Server provides the building blocks for implementing distributed environments in a way that provides universal access while preserving system performance. The platform is scalable at all levels, from the design of the data structures in the database through the deployment of server host machines to the distribution of sites in far-flung locales. The repository still appears to client applications as a unified whole despite the fact that its components are distributed across the globe.

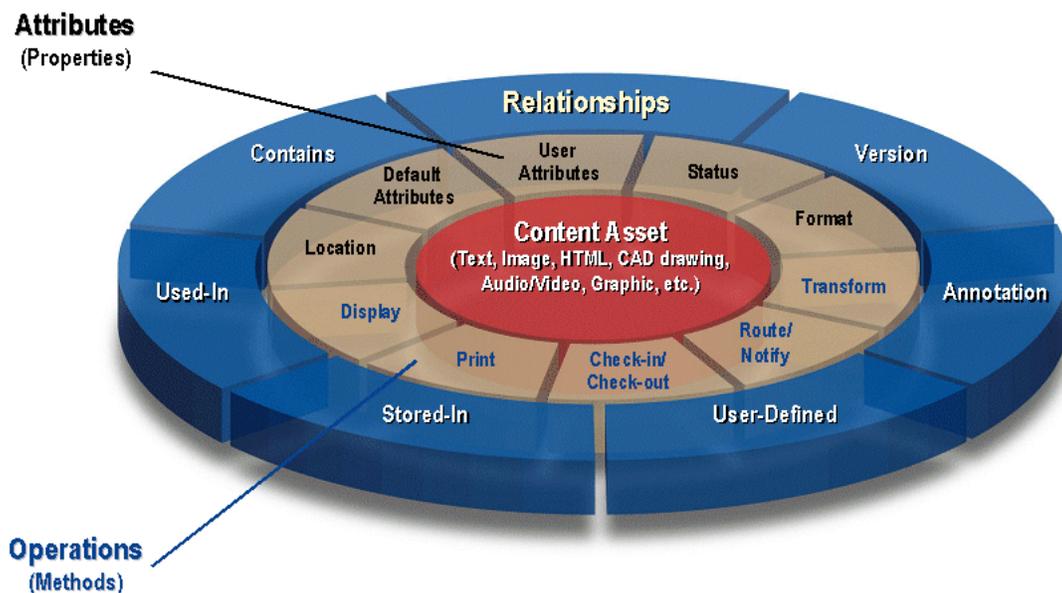
Figure 2-3. Globally Distributed Content Repository



Content Objects

In an object-based system like Documentum, an object is a component consisting of both data (content files, attributes, and relationships in the case of documents) and instructions for the operations available to be performed on that data (called *methods*). Just like the set of attributes, the set of methods for an object is configurable and extensible using Documentum development tools. Developers can create new object types that behave exactly as their specific business needs require.

Figure 2-4. Documentum Document Object



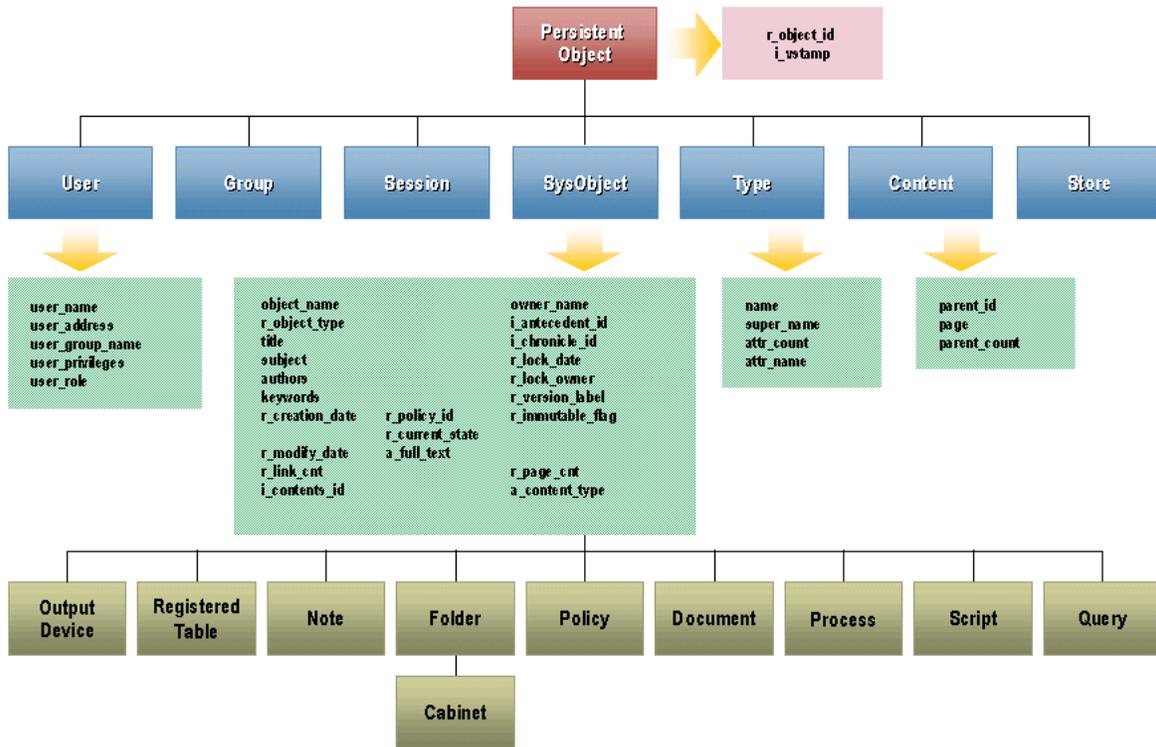
Client applications interact with document objects by calling their methods. From the client application's point of view, the methods are the same regardless of the format of the underlying content file. The encapsulation of the differences between formats enables content management applications to remain independent of formatting concerns.

Content Server is an object-based system: everything that users manipulate — documents, folders, security profiles, business processes, and so on — is stored and managed by Content Server as an object. Even the structural elements that define the repository itself are objects, including location objects, format objects, and relationship objects. The Documentum object model is the structure by which the server organizes the content and control mechanisms of repositories.

In object-based systems, object classes are organized into a hierarchy, which each object inheriting attributes and behavior (methods) from its "parent" in the tree. In the Documentum object hierarchy, the parent for most of the objects developers manipulate is SysObject. Objects derived from SysObject have SysObject attributes and methods as well as attributes and methods specific to that type of object.

All core enterprise content objects in a Documentum repository — documents, folders, cabinets, and others — are subtypes of SysObject. When this paper refers to a content object, it is referring to a SysObject.

Figure 2-5. Documentum Object Hierarchy and Sample Attributes



Content Server

The core services offered by Content Server can be divided into four categories, each of which builds on the previous categories:

- **Repository services**, which manage the data structures underlying the content and the repository itself
- **Library services**, which manage content objects
- **Core content management services**, which manage relationships between content objects in the repository, transforming them into units of business intelligence
- **Process automation services**, which manage changes made to the content as part of a business process

These services are available as methods of one or more Documentum object types.

Repository Services

The coordination of data from multiple sources into a content repository is an example of Content Server’s repository services. Content Server integrates the data in a way that ensures its integrity and security. It follows a transactional model, making sure not to commit changes in one place if it cannot make the corresponding changes in all places. The architecture ensures that Content Server is fault tolerant. Content Server also

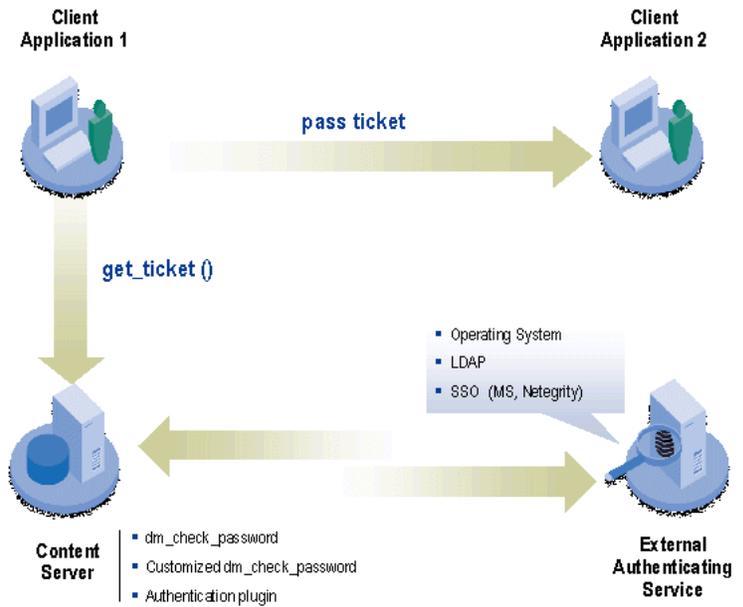
includes the tools necessary to administer the underlying data sources, including a complete administration application, auditing and tracking facilities, and replication and load balancing processes for distributed configurations.

Content Server stores and maintains a data dictionary, which is a collection of information about the repository and the objects contained within it. Applications can use the data dictionary to enforce business rules or provide assistance to users. For example, the data dictionary could specify that the value for a particular attribute must be unique, and applications can use that constraint when validating the data a user enters. Or, the data dictionary could contain a list of possible values (either a fixed list or a query returning a list) that applications can display to users as the available choices. The data dictionary can store error messages, help text, and user interface labels, all of which are available to content management applications.

Content applications use a name server called a *DocBroker* to identify the Content Server repositories that are available to it. When a client application wants to connect to a repository, the client contacts the DocBroker and requests the information it needs to connect with a server for the requested repository. The address of the DocBroker is identified in a configuration file (*dmcl.ini*) on the client machine. The DocBroker sends back the IP address for the host on which such a server resides and the port number that the server is using. If there are multiple servers for the repository, the DocBroker returns connection information for all of them, along with a proximity value indicating the server's relative distance from the client. The client session uses that information to choose a server and open the connection.

Data security is a critical consideration for enterprise content. Content Server requires a password for access to the repository, but does not store the password. It offers different and extensible options for how it authenticates users, such as contacting the operating system or an external directory via a Lightweight Directory Access Protocol (LDAP) service. If a user needs to open another session with the same repository, the application can use a *ticketed login* to avoid sending the user's password. When the first service call is made, the service requests that the server generates a login ticket and sends it back with the result message. The client then retrieves the ticket from the response and uses it in lieu of the password when establishing a new session with the repository.

Figure 2-6. User Authentication

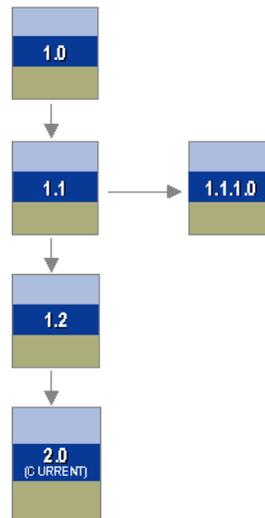


Library Services

Where repository services relate to the underlying structure of the repository, library services relate to the content objects. Library services transform the content repository into a library, controlling user access to each object in the repository. Every object has an associated access control list (ACL) or permission set that defines which users, groups, or roles can access the object and which operations they can perform. Content Server enforces seven levels of base object-level permissions and five extended object-level permissions. Organizations can define roles with specific application responsibilities and permissions.

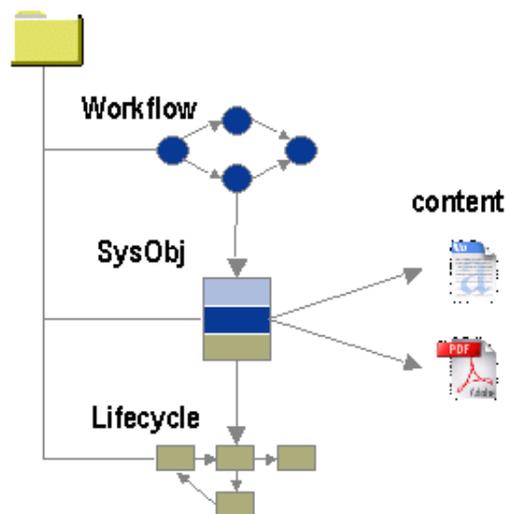
Content Server provides check-in and check-out services to ensure that users with permission to edit content do not overwrite each other's revisions or make incompatible updates. It also has automatic versioning capabilities for controlling, managing, and tracking multiple versions of the same content. Each distinct version is an object with its own version label. Content Server uses the version labels to create a "version tree" that captures the relationships between the versions.

Figure 2-7. Version Tree



Customers can establish many types of relationships between objects in the Documentum repository. Content Server includes several types of system-defined relationships, such as the relationship between a document and a note object representing annotations to the document or the relationship between a document and the workflow and lifecycles assigned to it. In addition, users can define custom relationships. For example, a relationship could be defined between two document types such that a document of one type is automatically updated when a document of the other type is updated. Developers can define this type of relationship and write procedures to manage it.

Figure 2-8. Object Relationships



Content Management Services

What constitutes a document, as a unit of content that is processed together, depends on the business process the content plays a part in. From a business perspective, a “document” may well consist of multiple components in different files or formats. Therefore, content management systems need to be able to treat multiple objects in the repository as part of the same overarching document. Virtual documents are an example where Content Server manages the relationship between objects so that they appear to client applications as a single entity. Content Server’s core content management services provide other examples: renditions, annotations, and version trees.

A rendition is an alternate representation of the content of a document, differing from the original in its format or its usage. Content Server can store content in the repository in any number of renditions; for example, a document might be stored as a Microsoft Word file, a Word file in Macintosh format, an Adobe Acrobat PDF file, and an HTML file with associated image files. A rich media file might have a full-length video rendition, a thumbnail rendition (storyboard), and a low-resolution rendition. Like versions, all renditions are treated as part of the same document. Content Server can automatically generate renditions of content using converters that are available through Documentum.

Annotations are comments that a user attaches to a document or other content. Throughout a document’s lifecycle, and often even after it has been published, people may need to record editorial suggestions and comments. The ability to attach comments to a document without modifying the original text is very helpful. Annotations are implemented as note objects, which Content Server associates with the content objects they relate to.

A feature of both content management and process management services, virtual documents are a way to link individual documents into one larger document. An individual document can belong to multiple virtual documents. When the individual documents change, the change appears in every virtual document that contains that document. (Content Server also supports *assemblies*, which are virtual documents built from specific versions of their component documents, so that future changes are *not* included.) Any or all of a virtual document’s contained documents can be assembled for publishing or perusal. Content Server can integrate the assembly and publishing services with popular commercial word processors and publishing tools. The assembly can be dynamically controlled by business rules.

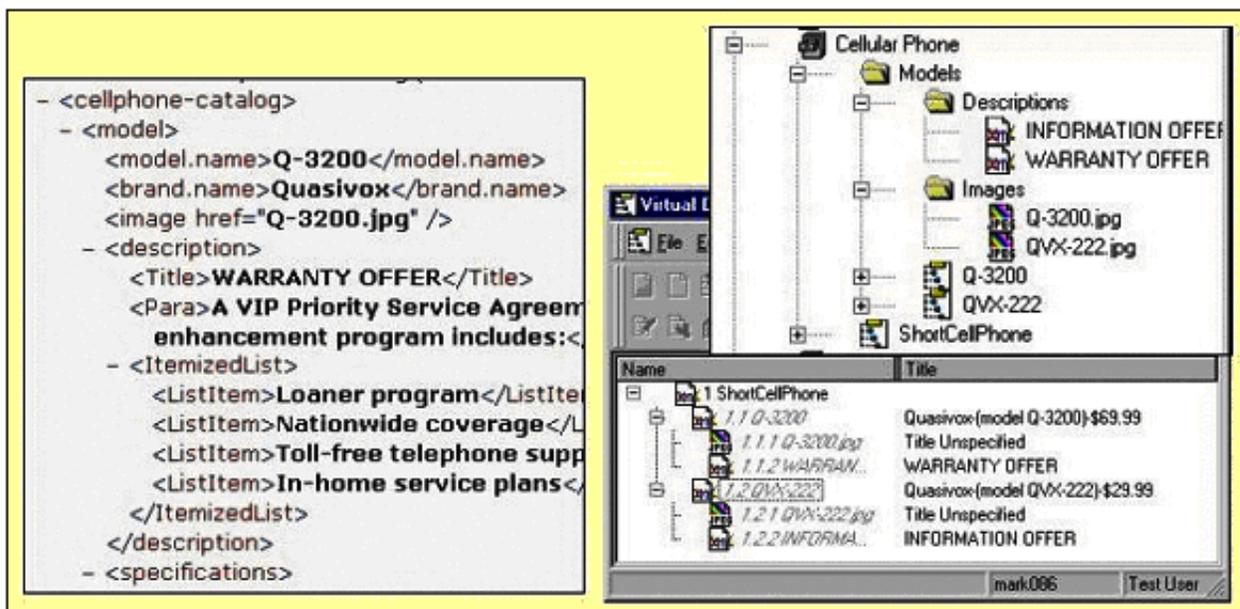
The core content management services enable companies to create rich repositories with an internal structure supporting the native structure of complex documents and logical organization of content based on actual business processes. To help client applications search for content within the repository, Content Server supports Document Query Language (DQL). DQL is a superset of SQL that provides a common, unified query language for all objects in the content repository. It enables you to search for content based on its attributes (including attributes it inherits from objects above it in the object hierarchy), the full text of its content files, its relationship to other repository objects, or a combination of these options. DQL respects the security of the content, returning only objects that the user is authorized to see. Content Server has a built-in query optimizer for best performance.

XML Content Management Services

Documentum provides a rich set of services for dealing natively with XML content throughout the content lifecycle. All standard Documentum services are available for XML content, as well as special XML processing.

XML is used for many purposes, and Documentum enables customers to create different XML schemas to represent different document types such as contracts, technical manuals, and catalogs. Each document type is likely to require distinct processing based on their required business behavior. Documentum provides a mechanism known as *XML applications* that allows highly configurable processing of XML documents of different types. There are no limitations to how many different XML applications an implementation can have.

Figure 2-9. XML Content Management



Documentum Application Builder provides interfaces for creating XML applications without any coding. Configurable aspects of an XML application include:

- Validation requirements, which determine whether the XML content is checked for well-formedness or is validated against a schema or document type definition (DTD)
- “Chunking” rules, which control how XML documents are divided into reusable component objects and stored as virtual documents
- Link recognition and parsing, which enables the application to recognize linking constructs in XML content and to automatically import, manage, and patch linked resources
- Handling of parsed entities, ndata entities, and base64-encoded content
- Automatically applying Documentum management features to XML “chunks,” such as assigning object types, setting security, creating folders, populating metadata, or assigning lifecycles

Documentum XML applications can also store and manage related XML files such as stylesheets, DTDs, schemas, editor stylesheets, and customizations. Documentum manages the distribution of these supporting files to clients as required.

XML applications work in conjunction with all of the Documentum library service operations. Whenever an XML file is checked in or checked out from the repository, Documentum invokes the XML processing necessary to parse or reconstruct the complete XML document. With Documentum's built-in XSLT engine, XSL transformations can be invoked directly from the repository, with the results stored back in the repository.

Documentum provides additional search capabilities for XML content. DQL has the ability to search within elements and attributes in XML content. XDQL is an XML interface to the DQL Query interface that returns search results as XML streams. XDQL also supports applying XSL stylesheets to returned results. One common use of XDQL is calling it from within XSL stylesheets, enabling the transformation to dynamically retrieve content and metadata from the repository.

Process Automation Services

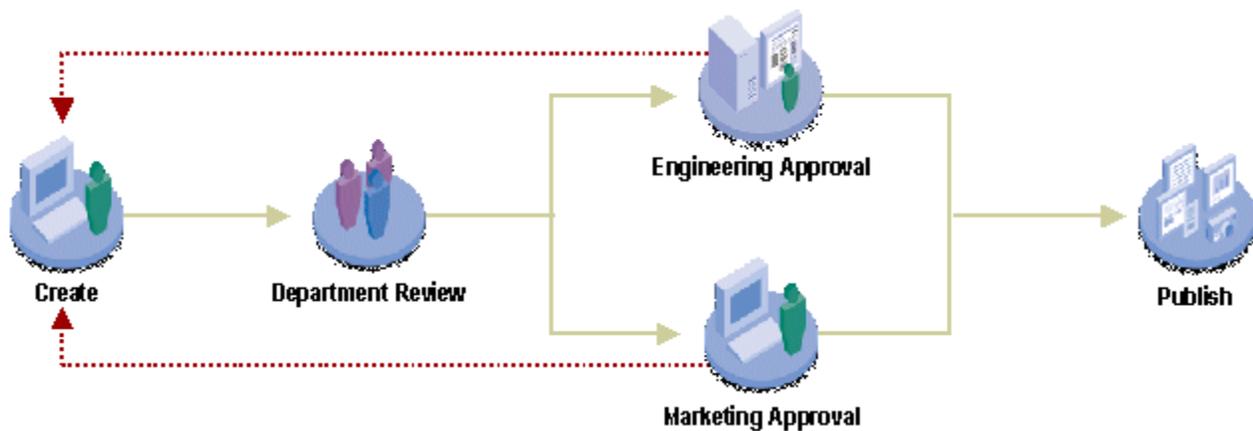
The process automation services provided by Content Server allow companies to define and enforce business rules and policies while users create, manipulate, or access content. They promote streamlining and optimization of business processes. The primary process automation features of Content Server are workflows and lifecycles.

A workflow formalizes and automates a business process, such as an insurance claims process or an engineering development process. The model supports both production and ad hoc workflows, enabling customers to manage the flow of information and tasks within and beyond the enterprise. Workflows can formalize long-lived business processes spanning days, months, or years. They can apply to individual documents, groups of documents, and virtual documents. Like all Content Server elements, workflow definitions are stored as reusable objects. After the business process is formalized in a workflow definition, users can use the definition to repeatedly perform the business process with different content.

Because a workflow's definition is separate from its runtime instantiation, multiple workflows based on the same definition can be run concurrently. Workflows can describe simple or complex business processes. A workflow can be serial, with activities occurring one after another, or parallel, with all activities happening at the same time, or can combine serial and parallel activity sequences. It can include explicit flows for exception or rejection cases as well as normal "forward" progress through the business process. The workflow engine determines the path that content takes through the flow by evaluating data-driven transition conditions after each activity.

Activities can generate work items in the Inboxes of the activity's designated performers. The designated performers can be specific users or roles that are dynamically resolved at runtime, such as the manager of the user who initiated the workflow. Work items represent work to be performed on the objects being routed through the workflow. Users with appropriate permissions can modify in-progress workflows and dynamically change the workflow routing. Workflow and event notifications are automatically issued through standard electronic mail systems while documents remain under secure server control in the repository.

Figure 2-10. Documentum Workflow



Many content assets within an enterprise have a recognizable life cycle. A document is created, often through a defined process of authoring and review, and then is used and ultimately archived or discarded. Documentum Content Server's lifecycle management services automate the stages in a document's life. A lifecycle object identifies a set of states that define the stages in a document's life. A change from one state to another is governed by business rules. The rules are implemented as requirements that the object must meet to enter a state, actions to be performed on entering a state, and actions to be performed after entering a state. Change of a lifecycle state can also be associated with an automatic change of document's access control, trigger workflows, generate a rendition, modify the document's attributes, or change its physical location.

For example, a typical lifecycle for Standard Operating Procedure (SOP) document has the states draft, review, rewrite, approved, and obsolete. Before an SOP can move from the rewrite state to the approved state, business rules may require the SOP to be signed off by a company vice president and converted to HTML format, for publishing on a company Web site. After the SOP enters the approved state, an action can send an e-mail message to the employees informing them of the SOP's availability.

Lifecycles identify *states* that documents can be in; they do not define when or how a document is promoted into each state. Workflows, on the other hand, are active. The defined business process is a connected network of activities, including information about who performs each activity and when each activity is performed. A document can participate in multiple workflows at the same time, but can have only one lifecycle.

Workflows and lifecycles automate business processes and policies related to specific content within the repository. Content Server also supports the automation of other required processes, such as administration tasks. Agents can be created to run jobs on a defined schedule. The jobs run in the background and generate trace logs that record their activity.

Extended Services

Documentum extended services products extend the capabilities of Content Server beyond its core content management services. They can extend any aspect of the content management framework: the content files (Media Services), the attributes (Content Intelligence Services), or the content capture and delivery (Content Exchange Services and Site Delivery Services).

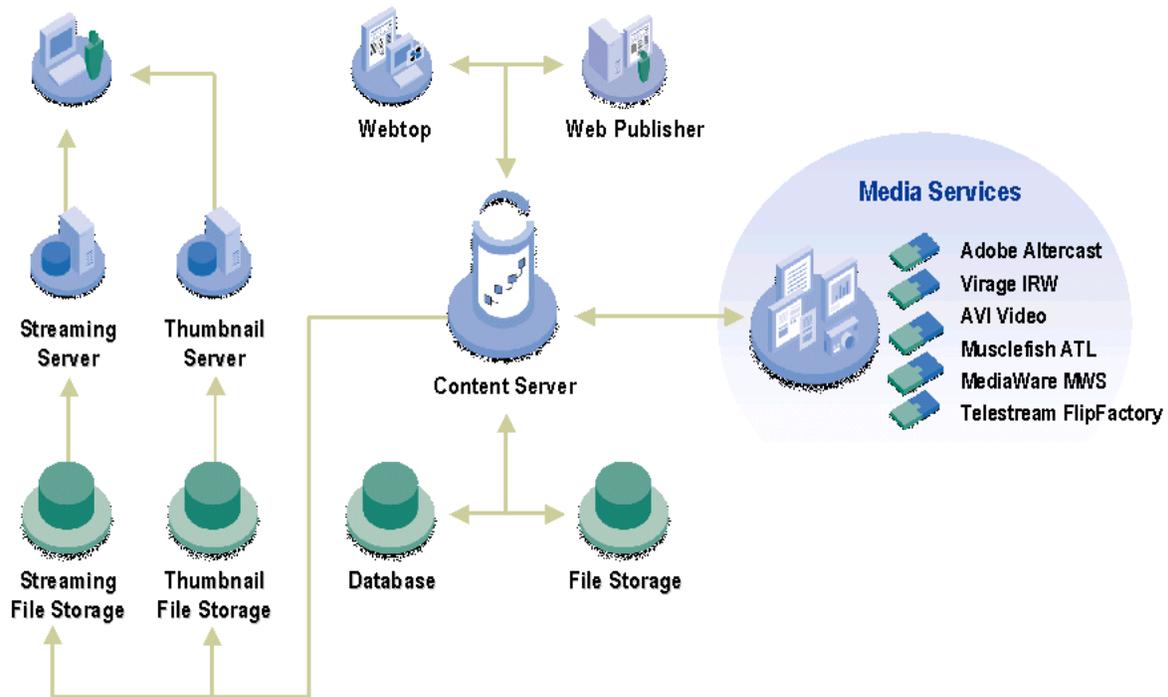
Media Services

Media Services is server software that integrates with Content Server to allow Documentum to convert content into other file formats. Media Services provides value-added management capabilities for specific rich media formats such as images, audio, video, and complex data formats of specialized applications including Quark QuarkXPress, Adobe InDesign, and Microsoft PowerPoint.

Media Services includes the following major components:

- Media Server, which provides the framework for file format analysis, property extraction, thumbnail creation, and transformation. It provides robust queuing and application monitoring of Media Plug-ins.
- Media Plug-Ins implement file format-specific capabilities to identify and extract properties, such as height, width, color mode, and compression, and generate thumbnails and low-resolution renditions of media objects. The plug-ins can convert files from one format to another (for example, from TIFF to JPEG, or from WAV to MP3). Plug-ins also provide users with the ability to perform transformations such as resizing, flipping, or rotating an image.
- A software development kit (SDK) allows customers and system integrators to create additional plug-ins for other formats, to extend the Media Services capabilities for specific solutions.
- Thumbnail Server, which is a dedicated server for secure, high-performance delivery of thumbnails directly to a browser from a Documentum repository. Applications retrieve the thumbnails from the repository as needed. The thumbnails are stored in file stores that are defined as the thumbnail storage areas.
- Streaming Server integration provides low-latency delivery of video and audio files directly from a Documentum repository. Third-party streaming servers integrate directly to Content Server file stores.

Figure 2-11. Media Services Architecture



Documentum provides plug-ins for most commonly used file formats and transformations, such as:

- Transforming images into other formats
- Performing image manipulation such as cropping, changing dimensions, or changing the color profile
- Transcoding video into Web-ready formats
- Applying optical character recognition to scanned documents and converting them into full-text indexed PDF files
- Building a new PowerPoint presentation from individual slides contained within other presentations already in the repository

The capabilities of Media Services can be accessed in a number of ways. A user may request a transformation on demand through a Documentum client, or a workflow task can request a transformation to include automated conversion of content as part of a business process. When a user adds a rich media object to the repository, Content Server recognizes its format as rich media. Content Server adds a processing request into the queue for Media Services. The Media Server retrieves the request and sends it to the appropriate plug-in for processing. The plug-in generates media properties, thumbnails, low-resolution renditions, generates storyboards if applicable, and performs transformations if requested. When the plug-in processing is complete, the Media Server updates the original objects, adding attributes and alternate renditions if necessary. The results of transformations are stored back into the repository, either as a new rendition of the source object or as a brand new object. When a new object is created, Documentum automatically creates a relationship between the source and derived objects, to allow the user or an application to track how and when the derived object was created.

Content Server stores thumbnails in a special file store that is shared with the Thumbnail Server. The Thumbnail Server is a server that uses Java servlets to manage thumbnail representations and HTTP technology to accelerate the display of thumbnail images in Web client applications.

With streaming server integration, when content in a streaming media format is checked into the repository, Content Server recognizes the format and figures out how the object should be processed and where it should be stored. The Content Server stores the streaming media in a separate file store from which the media can be streamed directly to the client.

Content Intelligence Services

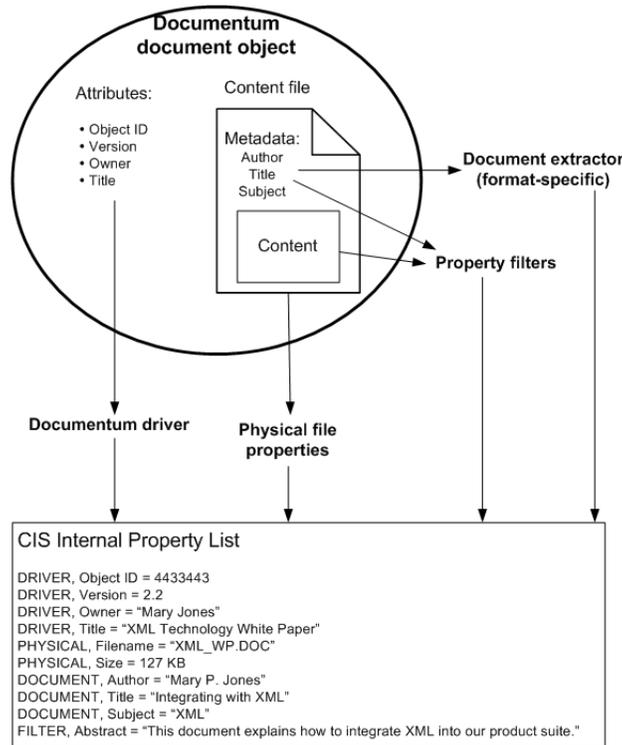
Content Intelligence Services (CIS) uses an independent server to analyze documents in a Documentum repository and extract a list of properties that describes each document. These properties can describe any aspect of the document for which CIS can extract or derive a value: its physical characteristics (such as file size or format), its repository attributes (such as object type or version number), metatags from inside the document (such as title or author), or custom properties. CIS performs a semantic analysis that determines what each document is about, using a customized taxonomy that lists the concepts to look for and identifies the evidence for the concepts.

When CIS is done analyzing a document, it has a list of the concepts discussed in that document. It can use any of the information it extracts in order to automatically set document attributes or link the document into appropriate locations in the repository. Extracting information from the document content and adding it to the document's attributes transforms unstructured data into searchable structured data. Because CIS adds attributes programmatically, they make consistent use of a standard vocabulary.

Content Intelligence Services goes through several steps to analyze a document it processes:

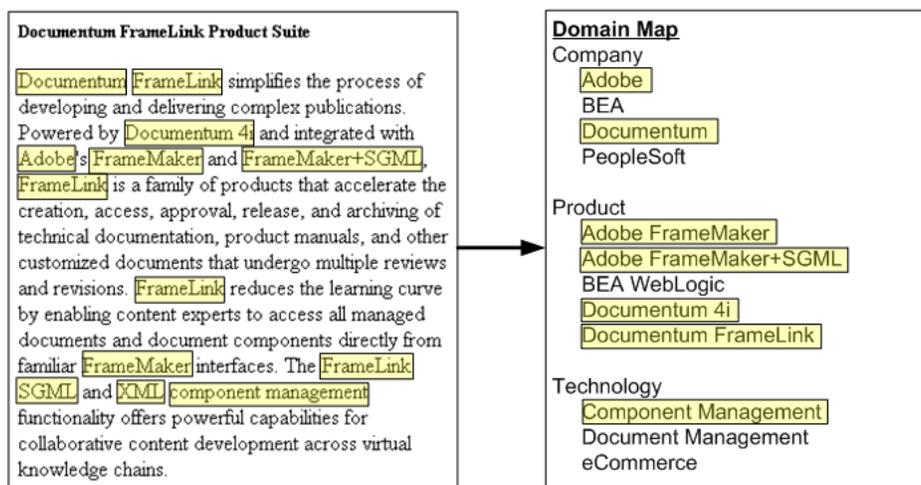
1. CIS retrieves a document from the Documentum repository. Based on information from the physical properties of the file itself, it constructs a list of property-value pairs describing the document.
2. CIS determines the file format of the document, then uses an extractor for that format to extract additional metadata properties and add them to the list of property-value pairs.

Figure 2-12. Extracting Information From a Document With Content Intelligence Services



3. CIS breaks up the document content into individual paragraphs and converts it into a normalized XML format.
4. CIS uses the taxonomy definition to perform a conceptual analysis of the XML-formatted document content. Based on the evidence it finds in the document, the server creates the concept list.

Figure 2-13. Conceptual Classification



5. CIS updates the repository based on the results of the analysis, setting document attributes, linking the document to new locations in the repository, or both.

Content Exchange Services

Content Exchange Services automates content exchange and cross-enterprise collaboration. It consists of three major areas:

- Content Aggregation Services — Collecting content from multiple sources, normalizing it, and storing it in the central content repository
- Content Distribution Services — Distributing content from the repository to multiple subscribers based on business rules
- Inter-Enterprise Workflow Services — Integrating people external to the organization into a business process involving content from the repository

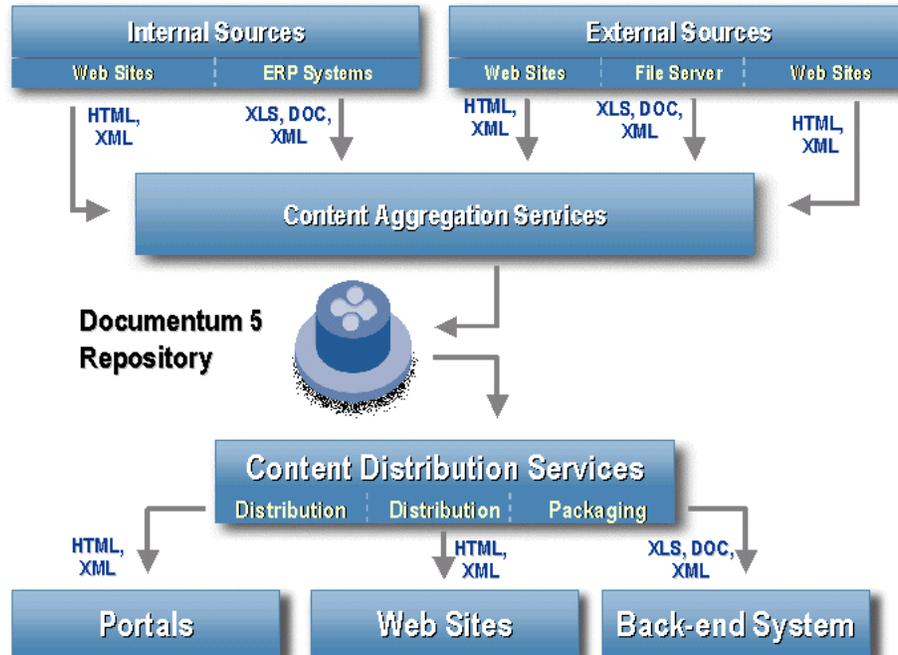
Note: In Documentum 5.2, the features of Content Exchange Services are provided through Documentum Content Distribution Services and Documentum Inter-Enterprise Workflow Services. The content aggregation features are not yet generally available.

Content aggregation involves a scheduled agent that searches possible sources for relevant content. It can retrieve content from a variety of sources, such as file systems, Web sites, or databases. When the aggregation agent finds content that needs to be added to the repository, it follows these steps:

1. It retrieves the content using standard protocols such as HTTP, ICE, or SOAP.
2. It determines the format of the content and performs any transformations necessary to prepare it for the Documentum repository. The set of available transformations is configurable and extensible.
3. It adds the new content to the appropriate location in the repository.

Once the content is saved in the repository, all of the content management services are available for processing it. In many cases, the new content has a workflow process associated with it.

Figure 2-14. Content Exchange Services



Documentum Content Distribution Services has two major components: a *Content Distribution Services server (CDS server)* and multiple *subscribers*. CDS server monitors one or more content repositories for updates, packages the new or updated content, and distributes it to subscribers. The server also manages the list of subscribers: their privileges, which content each subscriber is interested in, and how and where to deliver the updates. The subscriber accepts content packages from CDS server and updates its local content repository as appropriate.

A CDS server serves any number of subscribers. The server can offer content from Documentum repositories or Site Caching Services repositories, from relational databases, or from file-based repositories.

Subscribers can receive content using one of three delivery protocols:

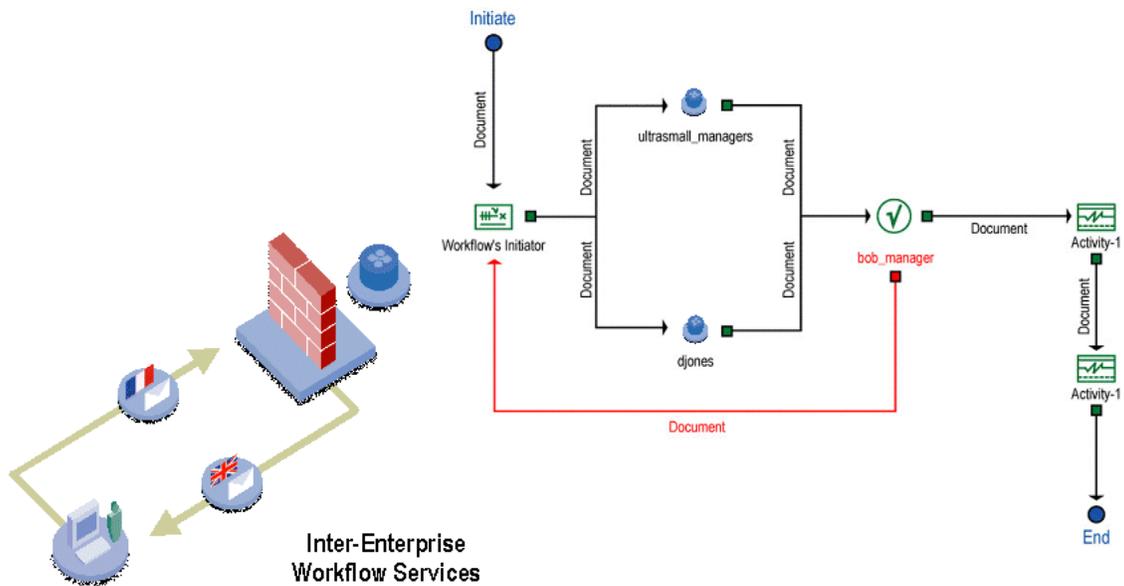
- Information and Content Exchange (ICE), an industry-standard message protocol for content distribution
- File Transfer Protocol (FTP), the Internet standard for simple file transfers
- E-mail using Internet standard Simple Mail Transfer Protocol (SMTP)

Documentum Content Distribution Services includes ICE-compliant CDS client software that a distributor can distribute to its subscribers. However, subscribers need not use the CDS client software to receive content through FTP or e-mail.

A content distributor may want to modify or enhance the content before delivering it to subscribers. For example, they might add a header to each document with their company logo or a date stamp. Content Distribution Services enable modification of the offer content using *templates*. A template contains instructions for manipulating offer content, which CDS server performs before sending the content to a subscriber. Content Distribution Services supports JSP templates and XSL transformations for XML documents.

Inter-Enterprise Workflow Services (IWS) is an ebXML-ready multi-protocol server that enables workflow processes to be shared with internal and external systems and individuals. With IWS, Documentum workflows can be extended across a company's firewall to include business partners. IWS also enables integration of Documentum workflows with workflow engines, including EAI and BPM systems as well as with workflows from other enterprise applications. It leverages open standards such as XML, SMTP, and HTTP/S to enable easy integration with other systems and to allow partners to exchange data over the Internet safely and securely. Partners participate in workflows when they receive and act on e-mail messages, or when e-mail triggers a workflow automatically at each partner's organization. When security is critical, Inter-Enterprise Workflow Services encrypts content through a secure socket layer (SSL) and supports digital certificates to authenticate users and content.

Figure 2-15. Inter-Enterprise Workflow Services



Site Delivery Services

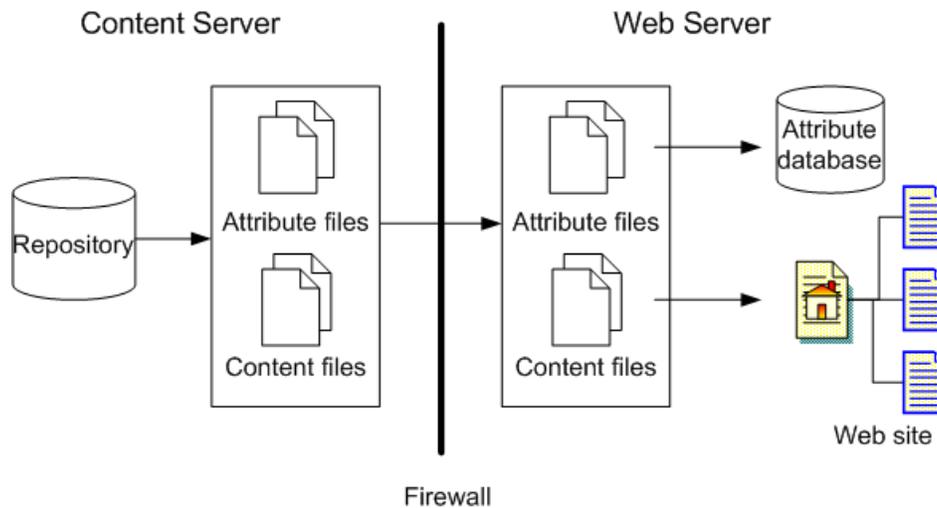
Documentum Site Delivery Services combines two Documentum products — Site Caching Services and Site Deployment Services — to publish content to multiple Web sites.

Documentum Site Caching Services provides a high performance repository for content delivery to Web applications. This repository serves as a cache for published content, allowing Web site managers to use the versioning, workflow, document lifecycle, and other content management capabilities of Content Server to maintain Web content. The Web site manager identifies groups of documents to publish to the Web, which version and format to publish, and when to publish them. Updates are staged in a way that prevents version inconsistencies.

In addition to publishing content files to a Web site, Site Caching Services can optionally export document attributes to a database on the Web server. Application servers can

access the attributes and provide application and personalization services based on the data. For example, they could insert the attributes into HTML metatags in the content files themselves, categorize the content based on their attributes, or personalize the content you display to different users. For example, a Web site for music lovers might classify documents by type of music. Classical music lovers see content related to symphony orchestras, while country music fans see content related to country singers and groups.

Figure 2-16. Publishing to a Web Site With Site Caching Services



The Site Caching Services software is composed of two separately installed components: source software and target software. The source software is installed on a Content Server host machine. The target software is installed on a Web site host. The target software can be installed on multiple hosts, allowing a single repository to support multiple Web sites.

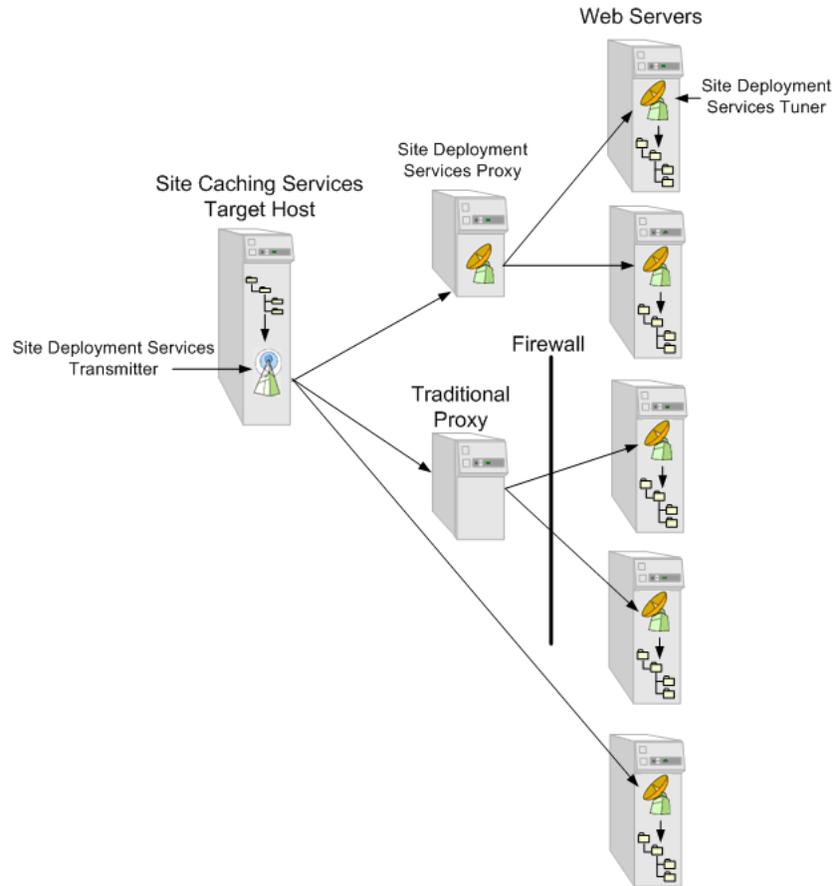
When publishing is initiated, either on demand or as part of an automated publication schedule, the Site Caching Services source software connects to the repository and locates the Web publishing configuration to use for publication. It queries the repository for the content it needs to publish, retrieving the content files and creating attribute files (in XML format) if that option is selected. The source software writes the files as an export data set on the file system of the Content Server host machine.

After the content files and attributes are exported from the repository, the source software connects to the Web server host as a transfer user. Site Caching Services can connect using the HTTP protocol, which does not encrypt data, or the HTTPS protocol, which encrypts data and is more secure. When the connection to the Web server host is established, Site Caching Services transfers the export data set to a directory on the Web server host, creating a Site Caching Services repository on the target machine. From here, the target software copies the content files to the Web site (in a directory structure that corresponds to the repository folder structure from which the files were published) and updates the attributes database on the Web server using the information from the attribute files.

The Site Caching Services repository on the Web server is available to other applications, not only the target Site Caching Services software. For example, another Documentum services offering, Site Deployment Services, retrieves the Web site from the Site Caching Services repository and deploys the site to multiple servers or Internet Service Providers.

Site Deployment Services can be configured to automatically deliver content on a scheduled basis, synchronizing deployment to any number of global Web servers.

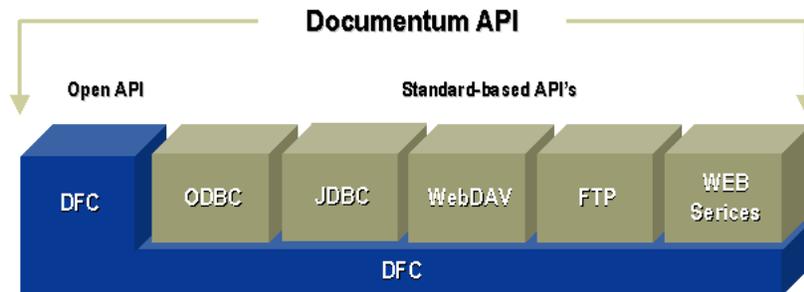
Figure 2-17. Site Deployment Services Builds on Site Caching Services



Interface Layer

Clients and applications use the interface layer to communicate with Content Server and interact with the content repository. The interface layer consists of Documentum Foundation Classes (DFC) and a number of standard interfaces built on top of DFC. Collectively, these products form the Documentum application programming interface (DAPI).

Figure 2-18. Documentum Application Programming Interface (DAPI)



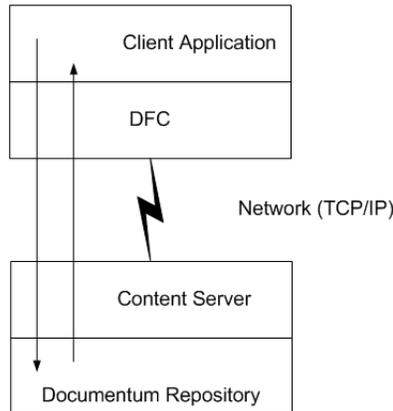
Documentum Foundation Classes

Documentum Foundation Classes (DFC) represents the richest API exposing all Documentum functionality. DFC provides an object-oriented framework for accessing the capabilities of Content Server. DFC exposes the Documentum object model as an object-oriented client library for content management applications to use. It is an API that can be used from a host of applications, including those developed in Java, Visual Basic, C#, and C++.

DFC is implemented as a set of Java classes and interfaces, along with a Java-COM bridge for accessing DFC via COM from Visual Basic or Visual C++.

Every computer running an application that accesses Content Server has a copy of the DFC software running on a Java Virtual Machine (JVM). The application on the client machine performs a method call through DFC, which translates the call into the Content Server's native API. Content Server instantiates the object, executes the method call, and returns the result to the client application.

Figure 2-19. Simple Client and Server Architecture



Applications access Content Server through DFC using a client object. Creating a client object loads the necessary shared libraries. The client object interface then serves as a factory for session objects, which represent connections to the repository. The application creates new repository objects or obtains references to existing objects through the session interface. Programmers familiar with the standard Java database connectivity package, JDBC, will see the similarities between that programming model and the DFC model.

Documentum Business Object Framework

Documentum also provides a way to extend DFC using the Documentum Business Objects Framework (BOF). Documentum Business Objects Framework provides a framework and a methodology to develop reusable business logic components called Business Objects. This framework is built into DFC and accessible from applications written using DFC. It is designed to provide the ability to develop pluggable components, each component implementing one or more middle-tier business rules. BOF can implement business logic in reusable business object components that can be plugged into middle-tier or client applications.

BOF enables programmers to develop highly reusable components that can be shared by multiple applications. For example, a catalog application would likely include a business object representing a product. The product business object has a number of attributes, such as the product name and SKU number, as well as a number of methods that implement the specific handling of products in the organization. For example, the Update method knows where product information is stored in the content repository and can keep it all synchronized. It can also maintain links to related information, such as the manufacturer data. All of these details are hidden from the application using the product business object. The business object provides a clearly defined high-level interface that the developer of a catalog application can easily understand. Once the product business object is created, it can be incorporated into other business objects such as a catalog object.

Figure 2-20. Product Business Object



There are two types of Documentum Business Objects, type-based business objects and service-based business objects. A type-based business object can extend a Content Server persistent object type and extend its capabilities by providing new methods for those types and allow overriding of existing methods. A type-based business object enables developers to define type-specific behavior without the need to customize each client type — the same business object can be used with Webtop- or Desktop-based applications. A service-based business object provides methods that perform more generalized procedures that are not usually bound to a specific object type or repository.

At any level, a method of a business object can be called by other DFC-based applications. JSP, ASP, Visual Basic, and other languages all have access to the business objects.

Web Services

DFC and BOF enable developers to encapsulate custom business logic that can be exposed as Web services. They provide a way to call functions on other computers across the intranet or World Wide Web. For example, a CRM system can communicate with Documentum through Web services. For Documentum customers, this means broader, industry standard access to content management functionality.

Developers can create their own services on our platform. Developers work with their preferred development environments, toolkits, and application servers to develop Web services over the Documentum API. Web services can be deployed on a Web server, where they are available to any networked client — a desktop application, a Web application or portal, or even another Web service or remote client/system.

For example, a BOF service could advertise its location so it can be found from remote locations across the Internet. One Web Services standard component, Universal Description, Discovery, and Integration (UDDI), provides the ability to register the service for discovery. Once the remote computer locates the service using UDDI, it sends a remote function call to the service using the Simple Object Access Protocol (SOAP). Unlike the binary protocols used with RPC and RMI, which are also remote function calling protocols, SOAP uses XML and HTTP to transfer parameters and return results in the form of SOAP Envelopes. The SOAP Envelopes were designed so that they can carry content in various forms.

For further details about Documentum support for Web services, refer to the technical white paper *Developing Web Services with Documentum*.

Standard-Based Interfaces

Documentum provides a number of standard interfaces in addition to the DFC API, simplifying access to the repository from authoring applications, application servers, and other components of the enterprise infrastructure. The standard interfaces include:

- ODBC and OLEDB — Many reporting tools, such as Crystal Reports and Microsoft Access, leverage Microsoft data access protocols for communicating with the Documentum repository using Documentum ODBC Reporting Services.
- JDBC — Many application server applications use the standard Java data access protocol to access content in the Documentum repository through Documentum JDBC Services.
- WebDAV — Documentum WebDAV Services provides a WebDAV server that enables WebDAV-aware applications, such as Adobe Photoshop and Documentum Desktop for Macintosh, to use this protocol to communicate with the Documentum repository.
- FTP — Documentum FTP Services is a FTP server for the Documentum repository that enables tools such as Macromedia Dreamweaver to integrate with the repository using the Internet-standard file transfer protocol.

Client Layer

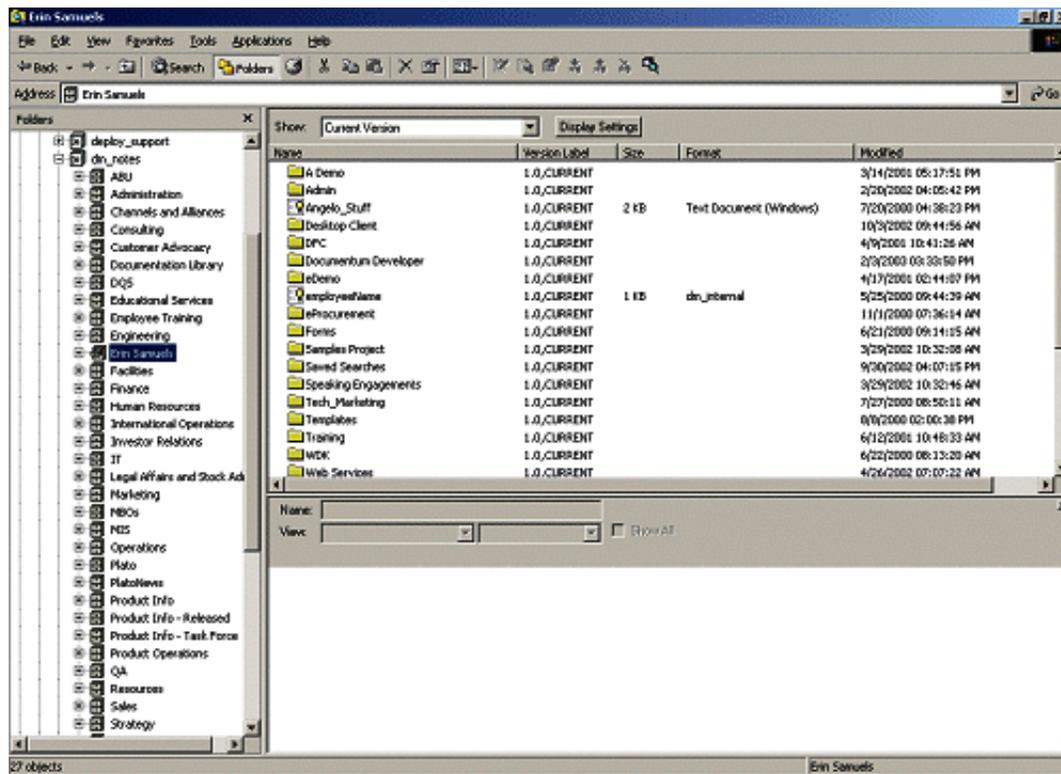
The client layer consists of basic applications for users to access the Documentum repository. It includes end-user interfaces, administrator tools, and integrations with popular authoring tools and enterprise applications.

Applications on the client layer are built from reusable components that build on the content management services available from lower layers in the architecture. A Documentum client application is a collection of components that work together to fulfill a business purpose. The same components may appear in other client applications or as portlets in an enterprise portal. Because components of both Microsoft Windows-based applications and Web-based applications share the repository's data dictionary and business objects, developers and users can be sure of consistent results from all client applications.

Microsoft Windows-Based Applications

Documentum Desktop is a Microsoft Windows application that gives end-users access to one or more Documentum repositories and exposes all enterprise document management capabilities. It is integrated with Windows Explorer, presenting repository contents in an interface parallel to Explorer's access to file-system contents.

Figure 2-21. Documentum Desktop



The functionality of Documentum Desktop is made available through COM components. Some of the components are written in Visual Basic, and the source code is provided. A workstation running Documentum Desktop has a copy of DFC, which the components use to communicate with one or more Content Servers.

Documentum Desktop also provides integration to popular third-party applications, such as Microsoft Office, to allow them to request content management services directly from the application's user interface. Documentum Desktop includes these integrations among others:

- Windows Explorer
- Microsoft Office
- Microsoft Outlook
- Acrobat Business Tools and Exchange
- Arbortext Epic and other XML editors

Developers can customize Documentum Desktop components or use the components in custom applications.

For activities that do not require immediate interaction with Content Server, Documentum Desktop offers the option to "work offline." This feature enables users to download content from the Documentum repository to their local machine and work on them while disconnected from the repository. When they reconnect to the repository, the system synchronizes the offline content with the matching objects in the repository. The synchronization process updates the repository with any changes the user has stored on the local machine.

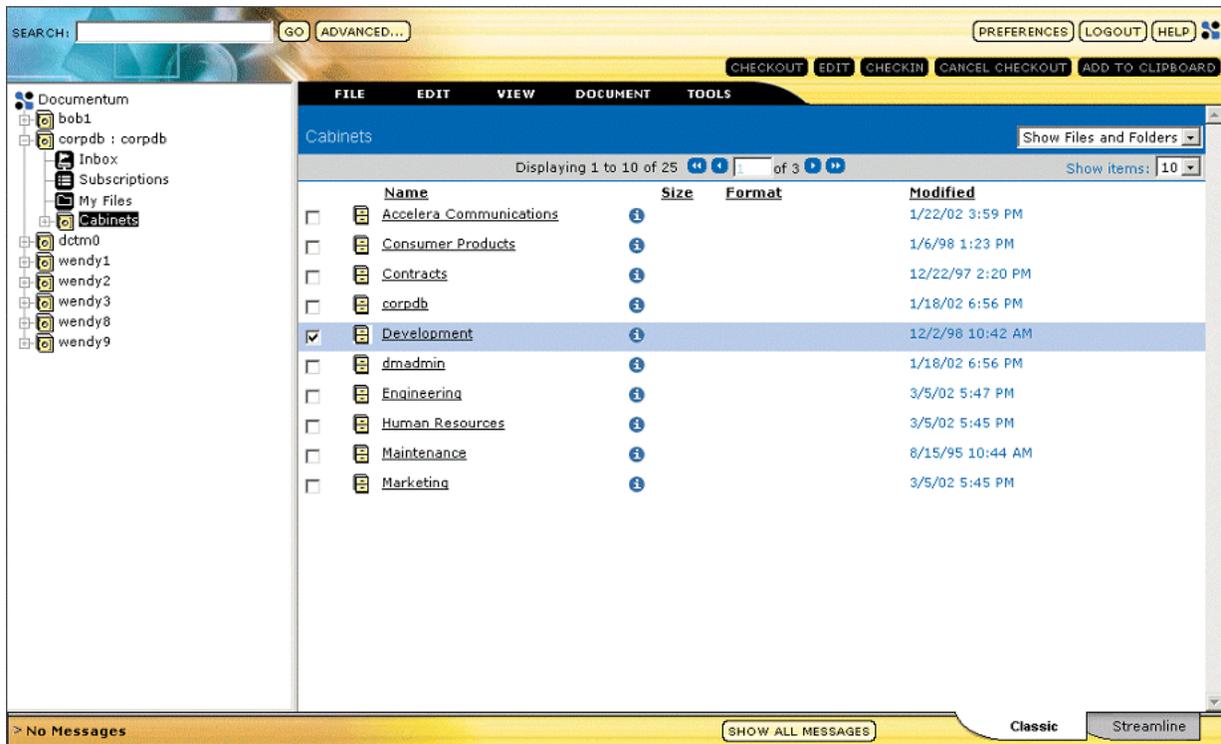
Authoring Integration Services (AIS) provides integration with content authoring tools to Documentum on both Macintosh and Windows platforms. AIS includes a server component that allows network drive access through a client's operating system. This server allows the operating system to treat the Documentum repository as a file system without compromising its security. From the authoring tools' point of view, they are reading and writing file from a disk. The AIS server is the primary access point for browsing, reading from and writing to the Documentum repository.

Additional client side development provides enhanced Documentum-specific features including import, check out, check in, metadata update, and version control. This development is achieved through the unified client framework. The unified client framework provides a mechanism in which client applications can interact with Documentum to utilize common content management functions such as metadata assignment, versioning, and lifecycle initiation.

Documentum Web-Based Applications

Documentum Webtop is a Web-based application that gives end-users access to one or more Documentum repositories. It provide a user environment similar to Documentum Desktop, running through a browser rather than Microsoft Windows Explorer.

Figure 2-22. Documentum Webtop



Webtop includes a branding service that enables customers to customize the look of their application user interfaces. The branding service manages the user interface appearance using themes, which incorporate images, icons, and cascading stylesheets. The user

interface controls, such as the OK button, label title, and View tab bar, can be configured with a cascading stylesheet style and the location of the image files used to render that control. Users can select from among the available themes. In multilingual environments, users can select their desired language for the user interface when they log in.

Webtop is built using the Documentum Web Development Kit (WDK). Just as developers can incorporate Documentum Desktop components in a Windows application (as delivered or in customized form), they can use WDK components in custom Web applications. The WDK uses a development approach based on a form-control-event approach, consistent with .NET WebForms and the developing Java Server Faces standard (JSR 127). This approach enables developers to create production-quality user interfaces far more quickly than traditional Web development techniques.

The WDK component model encapsulates customized and localized functionality into a package of closely related files, managed separately from the base product to ensure that the functionality is preserved during an upgrade. Any file — XML configuration file, user interface string resource, JSP layout file, Java class file, cascading stylesheet — can be copied into the custom area and modified. The files can be modified so that they inherit most of their behavior from the Documentum-delivered versions of the files, enabling them to override behavior where necessary without having to duplicate existing behavior.

Many key Documentum applications are built using WDK. For example, many Documentum products include a WDK-based administration module. Documentum Administrator, the core administration application for Content Server, is a WDK application. Documentum Web Publisher is a WDK-based application that enables customers to create, manage, and publish Web pages and Web sites using Documentum content management capabilities. Documentum Digital Asset Manager is a WDK-based tool for marketing departments or any other organization that deals with rich media.

Documentum WDK client applications access Content Server functionality in the same way as Documentum Desktop applications. The only difference with WDK applications is that the machine requesting services is the application server, not the end-user's workstation. WDK components, working with the application server, communicate with Content Server and pass the results to the user's browser.

Content Services Integration with Enterprise Applications

Documentum offers several products that enable interaction with the Documentum content management system from within other enterprise applications. Among the products are:

- Documentum Content Services for SAP
- Documentum Content Services for Siebel eBusiness Application
- Documentum Content Services for Lotus Notes Mail
- Documentum Content Services for Portals

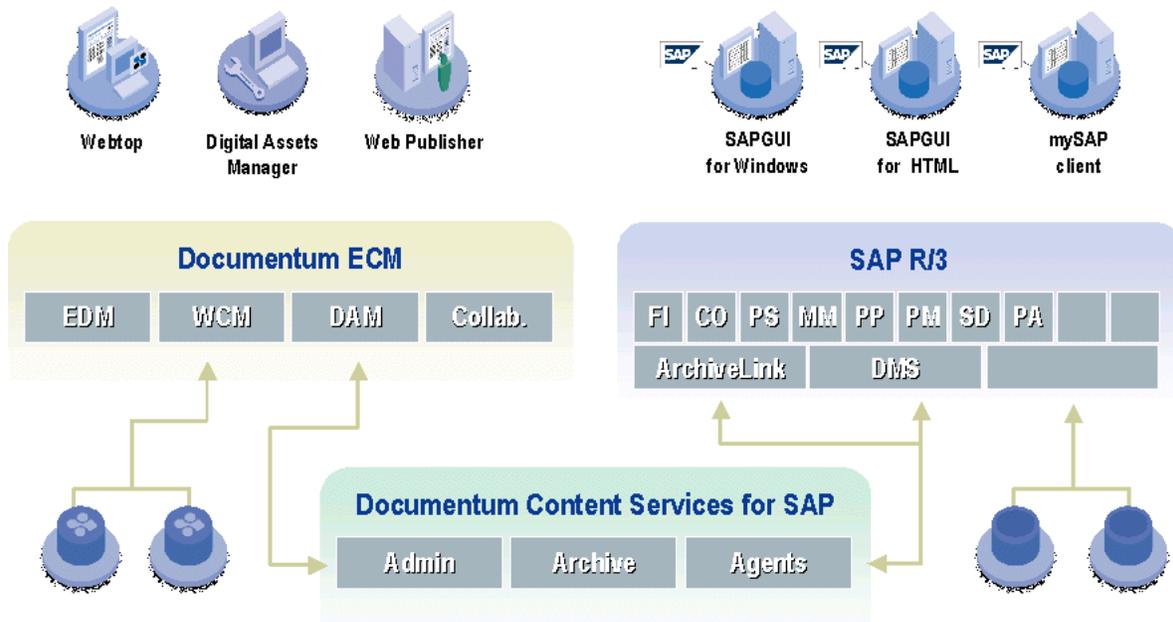
Also available are Content Services products for delivering content to various application servers, such as BEA WebLogic and IBM WebSphere.

Content Services links content in Documentum repositories with objects in the enterprise application, providing users with access to key Documentum functionality using the familiar interface of the enterprise application, eliminating the time and costs associated with searching for, filing, and storing documents. For example, an accounts payable clerk using an SAP application can instantly see a vendor’s contract, invoice, purchase requisition, and paid check from a single click on an SAP transaction report. Project managers can review standard operating procedures, material safety data sheets, engineering drawings, and specifications from within an SAP material master form.

Documentum enables rules-based processes for linking business content, triggering workflows, and notifying end users when relevant content has changed. Automating the linking process enables large volumes of scanned or remotely entered content, such as accounts payable invoices, to be easily linked to existing or new ERP records, ensuring error-free data entry and validation processes.

The Content Services products access the core content management functionality in the same way all other client applications do: through DFC and BOF.

Figure 2-23. Content Services for SAP



Developer Tools

The Documentum platform offers a number of ways for developers to access its content management functionality. It provides open access at each layer of the architecture, from direct low-level function calls to Content Server operations to pre-built presentation components that encapsulate complete user tasks. Developers can use Documentum development tools to customize existing client applications, incorporate Documentum-provided components into applications built with other development tools, or make calls directly to the DFC layer. Using the range of available tools, developers can quickly assemble content management applications from high-level components or exert fine control over basic system operations on the server.

The suite of developer tools from Documentum, called **Documentum Developer Studio**, includes:

- **Documentum Administrator** is a Web-based product for configuring Documentum content management systems, including implementation of system security.
- **Documentum Application Builder** is an integrated development environment for creating infrastructure objects for content management applications, such as custom document object types, workflows, and document lifecycles.
- **Documentum Foundation Classes (DFC)**, including the Documentum Business Objects Framework (BOF), is the object-oriented API for accessing core content management capabilities and encapsulating business logic in business objects.
- **Documentum JDBC Services** enables applications to access Documentum repositories using standard JDBC calls.
- **Documentum Web Development Kit (WDK)** provides a framework for developing Web-based content management applications.
- **Documentum Desktop Development Kit (DDK)** provides a set of tools and reusable components for building Windows-based content management applications.
- **Documentum Portal Integration Kit (PIK)** is a set of tools for integrating content management capabilities into a portal framework.
- **Documentum Media Services SDK** provides tools to extend server side content analysis and transformation capabilities.

This chapter discusses Documentum's tools for developing content applications and how they fit into the development life cycle.

The Development Life Cycle

The major steps in designing and building content applications are the same as for any enterprise software development project.

1. The first and most vital step is to analyze the business problem, identifying the application requirements, and designing the application components necessary to meet those requirements. Some development methodologies suggest applying more than half of the overall effort to this step. When the analysis and design are complete, specifications should be available that detail what Documentum objects and processes need to be configured and what custom code needs to be developed.
2. The next step is to configure the Documentum repository and servers. Documentum provides interactive tools that enable customers to structure the repository, create custom object types for special forms of content, define users, set security access for users and objects, and implement workflows and lifecycles for processing content.

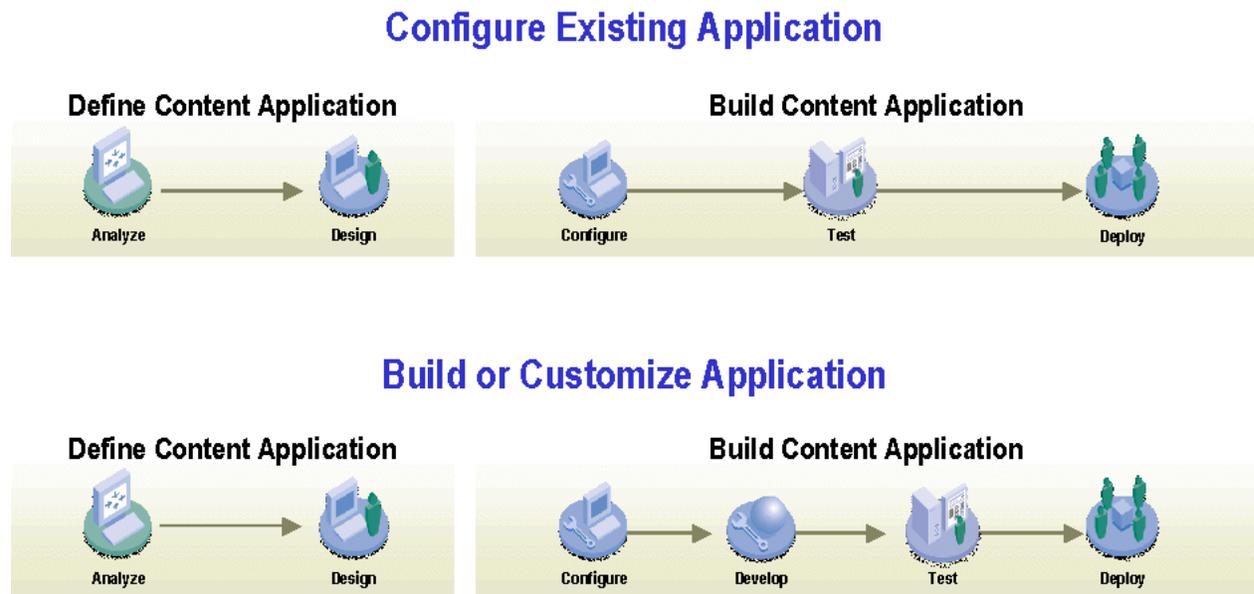
In implementations that use customized versions of delivered Documentum applications rather than developing new content applications, business analysts can perform these configuration tasks without assistance from developers.

3. Once the basic configuration is in place, development can proceed on creating or customizing content applications. The scope of the effort depends on how closely packaged Documentum applications meet the business needs and whether pre-built components are available for custom tasks. Documentum provides a variety of development tools for different levels of customization and different development environments.
4. Before putting any software project into production, it is important to test it. The test environment should match the production environment as much as possible. Documentum Application Builder enables developers to create packages containing all of the configured objects and processes for a project and install them into a test system repository. The packages, called *DocApps*, provide a straightforward way to transfer customizations between development, test, and production systems.

The results of initial testing typically require the project to loop back to one or more of the previous steps as issues and new requirements become apparent.

5. The final step is rolling out the software into production. DocApps and the Documentum DocApp Installer can transfer all customizations to the production system in a single package.

Figure 3-1. Development Life Cycle for Content Applications



Configuring Content Applications

It is perfectly possible to implement an enterprise content management solution using just applications provided by Documentum. The Documentum system provides all of the functionality necessary for managing content: users can create, delete, check in, check out, and version content in common file types such as Microsoft Word files, Excel spreadsheets, text files, and multimedia files. However, even when no significant development work is required, the server must be configured to fulfill the specific business requirements. It is necessary to define:

- Users and groups
- Permissions for those users and groups
- Object types that represent the organization's special categories of content
- Permissions for the custom objects
- A repository folder structure in which to organize content
- Automated tasks to manipulate and process content
- Document lifecycle stages through which content can move
- Task-based workflows through which to route content to people

The two Documentum products used to perform these application configuration tasks are Documentum Administrator (for the security-related tasks) and Documentum Application Builder (for creating custom object types and business process-related tasks). Documentum Administrator is a tool for general repository maintenance, and Documentum Application Builder is a tool for application development and deployment.

Documentum Administrator

Documentum Administrator enables administrators and developers to monitor, configure, and maintain Documentum repositories and servers throughout the enterprise. With Documentum Administrator, a system administrator can:

- Configure Documentum repositories
- Configure the secure access to repository content by creating users, groups, and access control lists
- Monitor system and resource usage
- Start and stop Content Servers
- Publish the contents of a repository

System administrators use Documentum Administrator once the Documentum system is in production. However, the first two items are important configuration tasks that developers or business analysts must perform as part of building the content application.

Documentum Application Builder

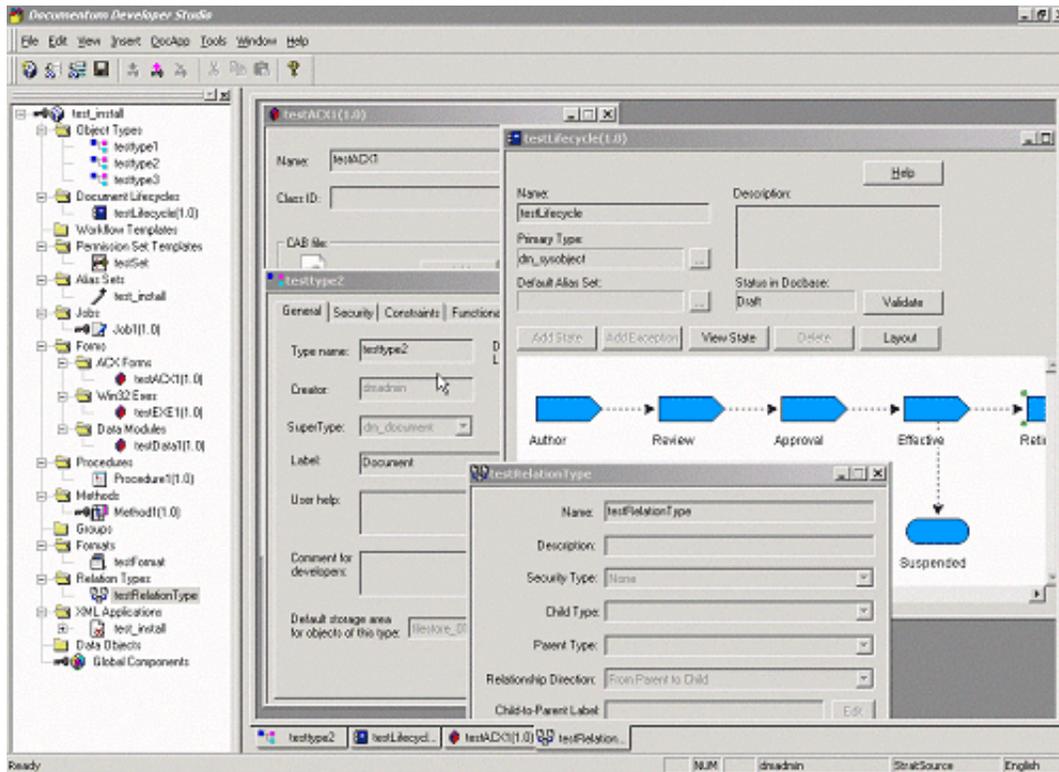
Documentum Application Builder is an integrated development environment for creating and maintaining most custom server application objects. Developers use Application Builder to develop custom object types, document lifecycles, workflow templates, permission set templates, alias sets, XML applications, relation types, formats, methods, procedures, and jobs. It also enables developers to specify constraints, validation, and value assistance (display pre-defined attribute values in client applications) for custom object types. It can package all of these customizations and components into a DocApp for easy transfer to other repositories.

Application Builder enables you to create and package:

- Object types
- Document lifecycles
- Workflows
- Permission set templates
- Alias sets (used to assign symbolic names to users, permission sets, or repository locations)
- Executable elements
- Data dictionary information
- Repository objects, such as queries or folder hierarchies

Application Builder consists of an integrated set of tools for creating each type of server application object. For example, it provides access to the Documentum Workflow Manager, a layout tool for designing and implementing automated business processes. With Workflow Manager, users can create re-usable workflows without having to manipulate the underlying objects with the application programming interface. Application Builder provides access to similar tools for creating document lifecycles, custom object types, and other server application objects.

Figure 3-2. Documentum Application Builder



Developing Content Applications

For implementations that require customizations that go beyond configuring server application objects, Documentum provides a range of development options.

From a developer's point of view, Documentum functionality can be divided into three layers: the presentation layer, the business logic layer, and the data model layer. The proper development tools for the job depend on which layer needs to be customized.

The *presentation layer* is used to display content and structured information in the Web browser or end-user application. The presentation layer consists of generic and custom user interface components that allow information to be displayed in different ways depending on such factors as the user role or the privileges. It roughly corresponds to the software that runs on the user workstation or, in a Web environment, the application server. The Documentum Desktop Development Kit (DDK) gives access to components for the presentation layer of Microsoft Windows applications. In a Web environment, Documentum Web Development Kit (WDK) would be the choice to present the layout and also manage the flow and sequence of user interactions.

The *business logic layer* implements the customer-specific business rules and policies. The business rules are built into common objects that can be reused in different application areas. The business logic layer is independent of the interface used to display the results; that is, it is independent of the presentation layer. If a procedure changes, the developer can modify the corresponding business logic component, and all applications are

automatically changed to the new policy. The Documentum Business Object Framework (BOF) is designed for creating business objects that encapsulate business rules.

The *data model layer* defines how business data is stored in the repository. It defines the content object hierarchy and the database schema. It is hidden from the presentation layer by the business logic layer, which fully depends on how data and content is organized and is therefore tightly coupled to the repository object model. As discussed in the previous section, developers use Documentum Application Builder (DAB) to create custom object types in the repository.

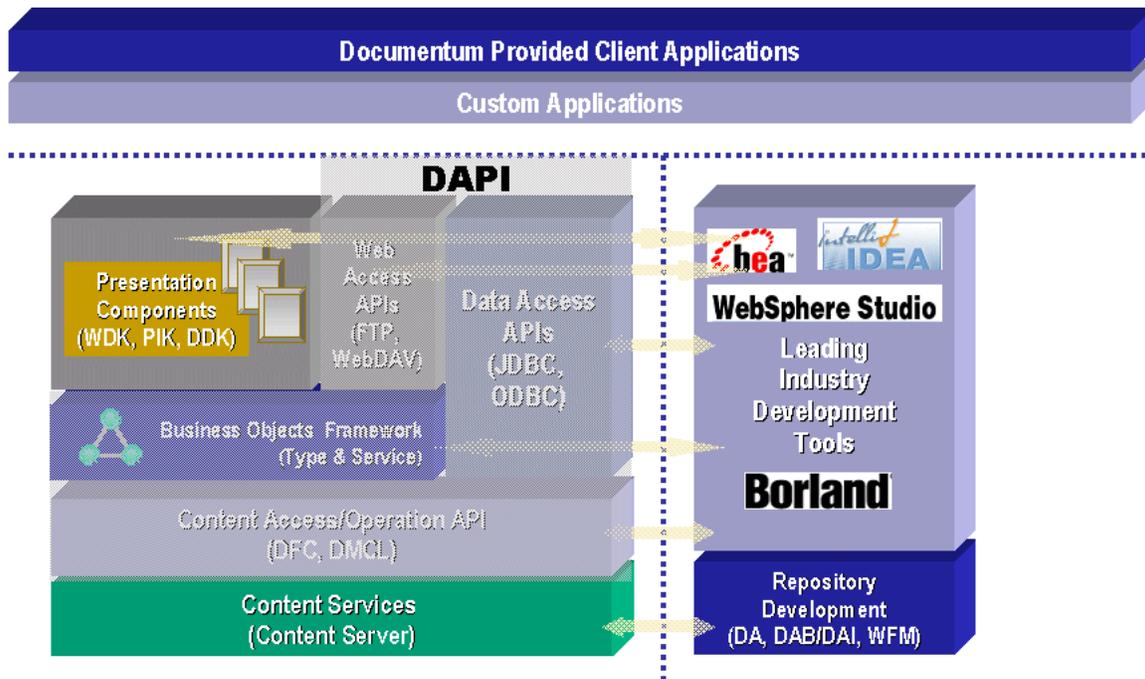
Implementing a complete content application typically involves a variety of coordinated customizations, for example, a custom document type (data model layer), a business object that controls the behavior of the type (business logic layer), and a user interface control for manipulating the type (presentation layer). The advantage of using Documentum-delivered components where possible is that all of the necessary objects and types have already been built.

Figure 3-3. Applications Involve Coordinated Customizations at All Levels



Figure 3-4, page 61 below shows how to determine which development framework to use depending on what functionality requires customization. Generally speaking, developers should target the highest-level layer that provides them with the level of control they need.

Figure 3-4. Documentum Development Tools

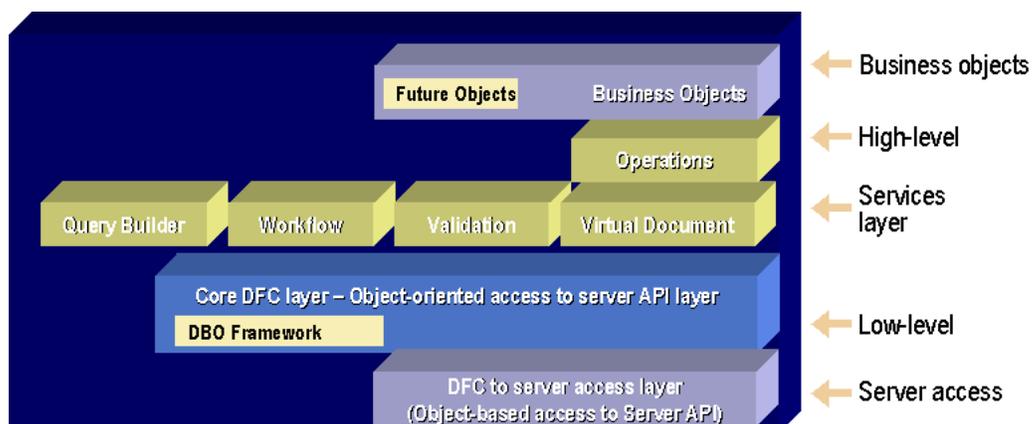


Documentum Foundation Classes

As described in Chapter 2, Documentum Foundation Classes (DFC) is the primary interface between content applications and Content Server. It provides an object-based application programming interface that fully exposes the Documentum object model.

DFC enables programmers to access content management functionality through both high-level operations and low-level object method calls. A services layer encapsulates business logic for common processes, such as workflow, data validation, searching, and virtual document management.

Figure 3-5. Documentum Foundation Classes



At its lowest level, DFC wraps calls to Content Server API functions. The Content Server API, referred to as DMCL (for Documentum Client Library), is a C library with interfaces to all server functionality. Documentum strongly discourages direct calls to the DMCL API. Since DMCL is written in C++, Java programmers would have to implement their own Java support layer. Also, DMCL does not support the higher level capabilities of DFC, such as virtual document management, Documentum business objects, and data validation. Documentum provides access to DMCL solely for compatibility with previous releases.

DFC consists of several packages, each containing the classes and interfaces for a particular range of functionality.

Table 3-1. DFC Packages

Package Name	Description
com.documentum.com	Interface for accessing DFC through OLE/COM
com.documentum.fc.client	Classes and interfaces for managing sessions and manipulating data in the Documentum repository
com.documentum.fc.client.common	Classes and interfaces to utility functionality associated with all DFC objects
com.documentum.operations	Interfaces to common high-level client functionality, such as checking in and checking out documents
com.documentum.registry	Classes and interfaces that manage Documentum information on the client's local system
com.documentum.xml.xdql	Classes and interfaces that enable querying the Documentum repository with results returned as XML

The sample code below shows the basic process of manipulating objects in the Documentum repository using DFC. The code fragment from a Java program creates a new document in the repository using a file from the local file system as its content.

```

IDfClient client = new DfClientX().getLocalClient();
IDfSessionManager sMgr = client.newSessionManager();
    IDfLoginInfo loginInfo = new DfLoginInfo();
        loginInfo.setUser( "Mary" );
        loginInfo.setPassword( "ganDalF" );
    sMgr.setIdentity( strDocbaseName, loginInfo );

IDfSession session = sMgr.getSession( strDocbaseName );
IDfDocument document = null;
    document = (IDfDocument)
        session.newObject( "dm_document" );
    document.setObjectName( "Report on Wizards" );
    document.setContentType( "crtxt" );
    document.setFile( "C:\Temp\Wiz.txt" );
    document.link("/DFCObjCab/DFCObjFolder");
    document.save();

finally {
    sMgr.release( session );
}

```

The major steps are:

1. **Establish a session with the Documentum repository.** To obtain a session, the application must instantiate an IDfClient object, obtain a session manager object from it, and provide login information for the repository. The session manager object creates the session, represented by the IDfSession object.
2. **Obtain or create the repository object to manipulate.** Methods of IDfSession object, such as newObject or getObjectByQualification, are used to get or create repository objects. In the sample, a document object is created using the method IDfSession.newObject. Note that the newObject method returns an IDfPersistentObject object. The sample code explicitly casts it to an IDfDocument object, then uses the document object's save method, a method that IDfDocument inherits from IDfPersistentObject.
3. **Manipulate the repository object.** Typically applications will use routines from the operations package to manipulate the object — for example, check it out, modify it, and check it back in to the repository. The sample code populates and saves the IDfDocument object by calling some of its low level methods to assign a name, type, and content file.
4. **Release the session.** When the session is released, control returns to the IDfSessionManager object. The IDfSessionManager object can reassign it the next time the application calls the newSession method.

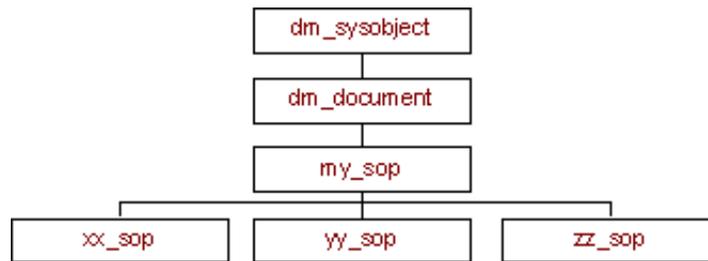
Most DFC methods report errors by throwing a DfException object. Java code like that in the above example normally appears within a try/catch/finally block, with an error handler in the catch block. Visual Basic code uses the On Error Goto statement to handle exceptions.

Business Objects Framework

Business objects are registered with DFC so that they are available from any code that uses DFC. The business object framework is a standard on which Documentum and its partners build business object components as extensions to the DFC. It provides a means to distribute components based on a standardized and published framework and thus eases distribution and integration of new services and objects.

The Documentum platform represents the structure of business logic using the object hierarchy in the repository. For example, suppose a company needs to store various different types of standard operating procedure (SOP) documents. The developer can create a generic SOP object as a child of the standard dm_document object, then create specialized instances of the SOP as children of this object. The children inherit the properties of the parent SOP object and can also have properties of their own.

Figure 3-6. Custom Object Types



This object-based model has always been one of the strengths of the Documentum architecture and can be seen in every Documentum installation worldwide.

Suppose that a company policy states that every time a new version of any SOP is checked in, a notification should be sent to the Director of SOPs. Without using business objects, this business logic would have to be written into every client application. The logic would be simple — if the object type is `my_sop`, send a notification after check in — but it could easily be broken if a client did not implement it or different clients implemented it in inconsistent ways.

Using business objects, the developer can attach the business logic to the objects in the repository. By creating a type-based business object, meaning a business object associated with a particular object type, and linking it to the `my_sop` object type, the notification becomes part of the object's standard behavior. The logic exists in the DFC so that any time an object of this type is checked into the repository, a notification is sent. When multiple client applications need to manipulate the same SOPs for distinct business purposes, they can share the same objects, making them automatically compatible with each other.

There are two types of business objects:

- Type-based business objects
- Service-based business objects

Type-Based Business Objects

Type-based business objects encapsulate behavior that is specifically bound to a particular object type in the repository. They are used to provide new behavior for object types, such as enforcing data validations or performing custom actions in response to system events.

All persistent object types in the Documentum repository (such as `dm_document` or `dm_user`) have a class or interface representing them in the DFC (`IDfDocument` or `IDfUser`). The interface allows developers to manipulate the objects; for example, a `dm_document` object is saved using `IDfDocument.save()`.

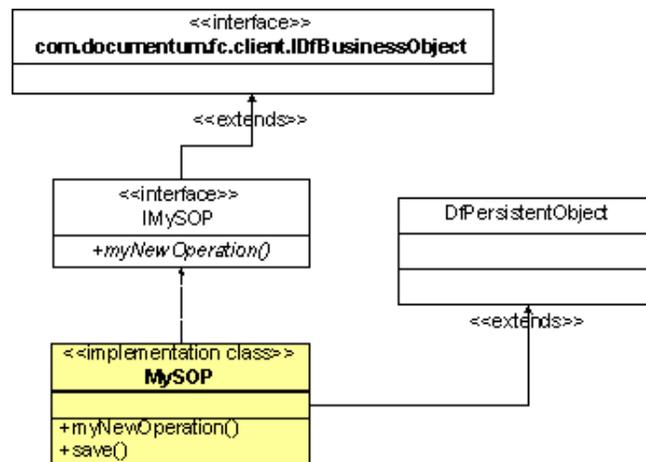
Type-based business objects enable developers to define their own interfaces for custom object types by defining a class in the DFC for them. This class extends `DfPersistentObject` or a subclass, enabling developers to override the behavior of existing DFC methods, or add custom methods. Thus, it is possible to have a type-based business object class that

represents an SOP so that `IMySOP.save()` invokes an SOP-specific implementation of the `save()` method, or `IMySOP.myAction()` invokes a completely new method.

A type-based business object consists of:

- A Java interface that contains the methods that can be performed on the object type's data
- A Java class that reflects an object type and contains implementation of the methods that can be performed on the object type's data

Figure 3-7. Type-Based Business Object



Service-Based Business Objects

Service-based business objects implement business logic that does not need to be associated with a specific object type. They are used to implement logic not directly related to objects, or logic shared between object types.

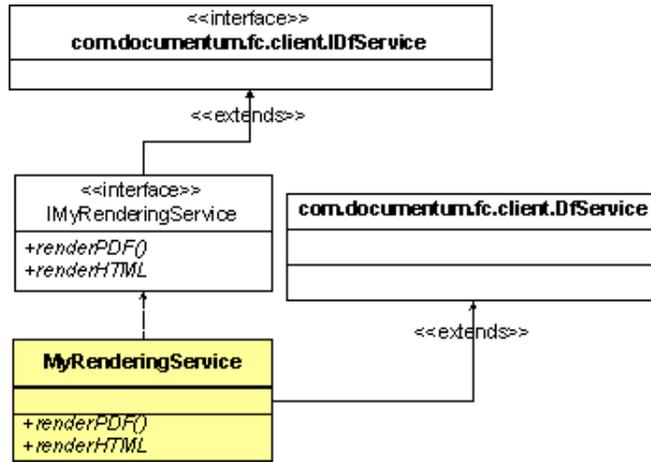
For example, a common requirement is to render HTML by applying an XML stylesheet to an XML file. One way of implementing this requirement is to create type-based business objects for each of the object types that may need to be rendered. This approach creates a great deal of redundancy. A better approach is to move the rendering logic into a service-based business object, then have each type-based business object call the service-based object to perform the rendering.

A service-based business object consists of:

- A Java interface that extends (subclasses) the `com.documentum.fc.client.IDfService` interface and defines the methods (services) that the service provides
- A Java class that implements this interface

A service-based business object does not have a corresponding object type.

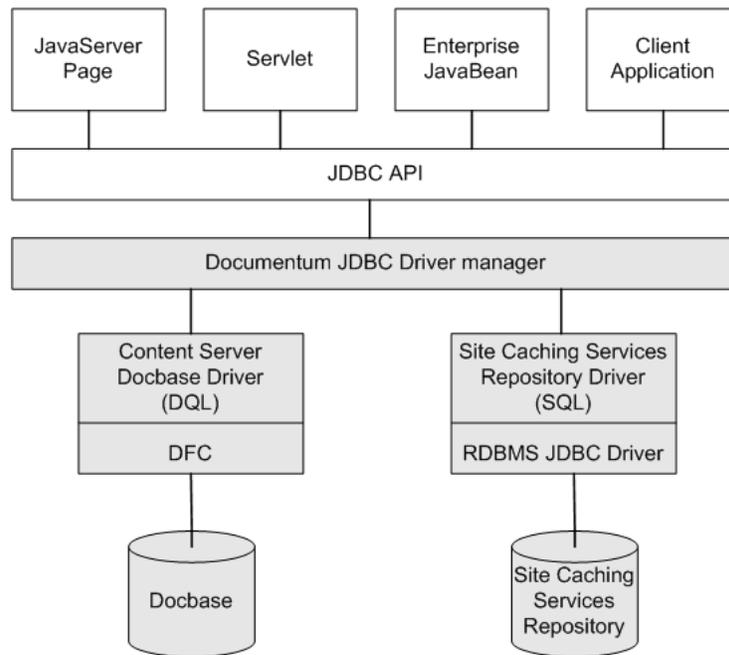
Figure 3-8. Service-Based Business Object



Data Access APIs

The data access APIs enable applications to access content in the Documentum repository using industry-standard interfaces. For example, Documentum JDBC Services enables applications to connect to Documentum repositories using the Java Database Connectivity (JDBC) interface. JDBC is an interface for accessing repositories from Java client applications, Java Server Pages (JSPs), or servlets. With Documentum JDBC Services, applications can connect to a Documentum repository, issue server API commands, perform DQL or SQL queries, access metadata, and retrieve content from the repository. When an application connects to a Documentum repository using JDBC Services, it implements the JDBC 2.0 API using Documentum's DQL instead of ANSI SQL.

Figure 3-9. Documentum JDBC Services



Documentum JDBC Services translates JDBC API calls into DFC calls so that an application can communicate with Content Server in the same way it communicates with databases or other repositories. Client applications can use existing database access code and work correctly with Documentum repositories.

JDBC Services supports most standard JDBC APIs. It implements the JDBC 2.0 extension API `javax.sql`. The extension API enables the use of data sources, which are general-purpose objects for specifying databases and other resources to applications. JDBC Services also supports the Java Naming and Directory Interface (JNDI), so that applications can connect to the repository using logical names rather than hard-coded names.

JDBC Services supports connection pooling. Connection pooling allows applications to use connections that exist in a pool rather than opening and closing new connections. This approach improves application performance because new connections do not have to be constantly created and destroyed.

Documentum Web Development Kit

The Documentum Web Development Kit (WDK) provides a framework on which to build Web applications that connect to the Content Server. Most Web-based content applications access Documentum content management capabilities through Documentum WDK. Applications built using WDK reside on an application server. Users access them through a browser, just like any Web application.

The WDK framework runs as a Web application on J2EE-compliant application servers. The WDK programming model is based on XML and J2EE technologies:

- A presentation layer uses HTML and JSP (JavaServer Page) tag libraries to control the Web page user interface and application configuration
- A component model that encapsulates a comprehensive set of Content Server functionality in configurable server-side components

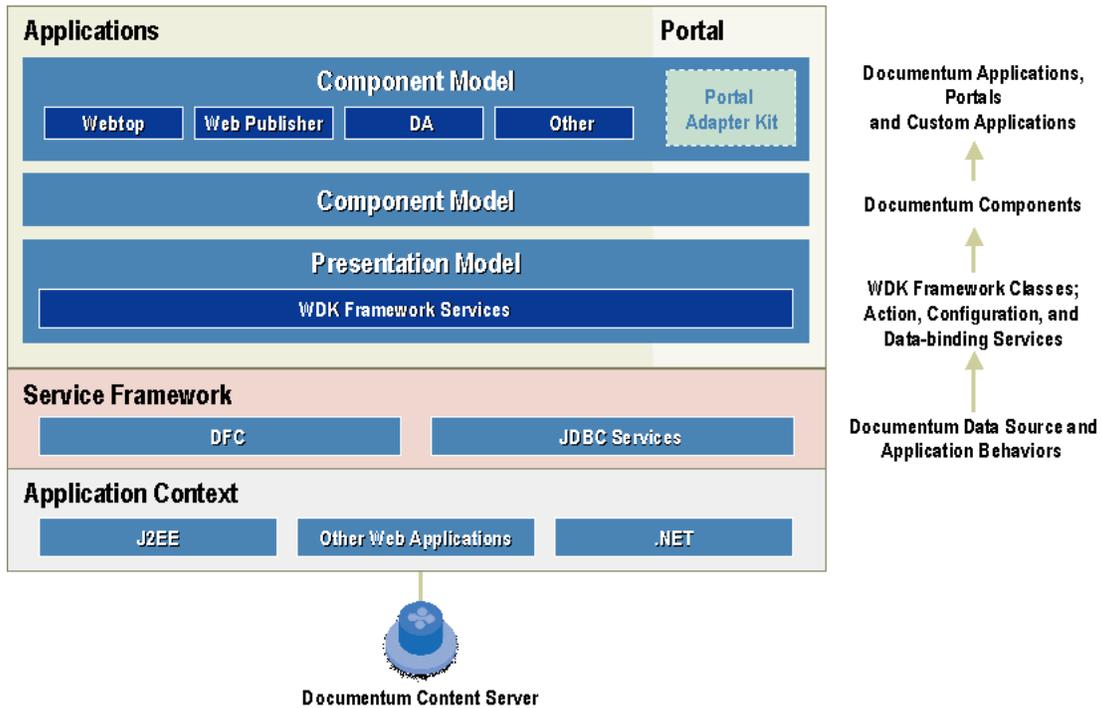
WDK also provides robust server-side state and browser history management for content applications. Components are configured through XML configuration files.

A JSP page in WDK consists of fixed (template) HTML and dynamic content rendered by JSP technology. Most of the application behavior is represented by JSP tags from Documentum tag libraries. The application logic is separate from the JSP page, in tag library classes. User interface events are translated into method calls to server-side objects. This approach separates page layout from application behavior and allows applications to reuse user interface elements (controls and forms) without affecting behavior.

JSP pages are compiled into servlets (Java classes) by the JSP container or by a third-party compiler. WDK provides JSP pages that are compiled into servlets. These servlets execute on the Java application server and either perform a server task or generate dynamic content that is then displayed on the client browser. WDK servlets conform to the Servlet 2.2 specification for Web applications, as do the JSP pages, client-side applets, and tag libraries.

The WDK-based application gets data from the Content Server via DFC, encapsulates the data in Java classes on the application server, and renders the data to the browser. Developers can use a DFC session interface, an in-memory recordset, or a JDBC connector as a data source.

Figure 3-10. Documentum Web Development Kit (WDK) Architecture



WDK architecture consists of these layers:

- The Java application server and J2EE provide the application context. WDK runs within a J2EE application server.
- The service framework provides content management capabilities, which are implemented by Documentum Foundation Classes (DFC) and by the Sun Java SDK.
- In addition to the control tags and behavior classes which influence the user interface, the presentation model incorporates capabilities related to browser management and application configuration.
- The component model provides a configurable, encapsulated set of Documentum functions or *components*. Component JSP pages use WDK controls from the tag libraries, and each component can handle a control event with its own event handlers.

The WDK framework enforces a contract for each component, consisting of parameters that initialize the component. The component behavior class includes events that respond to user action and properties that get and set the state of a component. The component contract is defined in an XML configuration file.

- The application layer consists of a set of components that are composed of JSP pages, supporting behavior classes, and XML configuration files. The JSP pages are modeled by form classes that manage state and navigation. The user interface in forms is modeled by controls, which are widgets represented by JSP tags. The application is supported by the WDK framework of services, such as the configuration, action, messaging, and tracing services.

A Web application can contain several WDK application layers. The application model enforces consistent appearance and behavior across all application layers contained within the root application. The J2EE-compliant root context can contain application layers that inherit application parameters from other application layers. The custom application layer can then extend the existing application layers.

Using the branding engine, an application layer can apply themes that provide the application's unique appearance through icons, images, and stylesheets.

The WDK framework provides for rapid development of Web applications that access Documentum repositories. Included with WDK are many pre-built components that are easily configurable and easily integrated into an application. Custom Web applications can be based on WDK or can extend other WDK client applications.

Documentum Desktop Development Kit

The client product Documentum Desktop is built from COM components that encapsulate content management functionality. For example, the Login Manager component displays a dialog box that prompts the user for a repository to log in to, their user name, and their password, connects the user to the repository, and manages their session. Developers can incorporate the Login Management component in their applications. In this way, they do not have to write the corresponding code from scratch.

Developers can use any COM-enabled development environment, such as Visual Basic or Visual C++, to build Windows-based clients that use Desktop components. Desktop component properties and methods are exposed to the integrated development

environment through standard COM interfaces. The Documentum Desktop Development Kit (DDK) includes the Visual Basic source code for many Desktop components so that they can be customized. In addition, developers can use DFC interfaces, classes, properties, and methods in their client applications. DFC is exposed to COM-based clients as a type library. (A type library is a file created using the Microsoft Interface Definition Library that contains type information about exposed objects — in this case, Documentum objects.)

The DDK consists of:

- Visual components — Components that display a user interface making an action, such as importing or checking in files, available to users
- Framework components — Components that implement the functionality underlying the visual components
- Component Dispatcher — Decides which component needs to be invoked depending on a variety of conditions; for example the Component Dispatcher could invoke a different Properties component depending on the user, lifecycle stage, or object type

Documentum supplies the Visual Basic source code for most visual components so that they can be customized. Developers can also replace them completely with new components. As long as they meet the standard interface requirement, newly developed components can use the Documentum Desktop framework and can be launched using the Component Dispatcher. The interface requirement is simply that the component must implement the COM interface IDcComponent with three methods: Init, Run, and DeInit.

Desktop components are standard COM components and all implement the Init, Run and DeInit methods. Although it is possible to invoke these methods directly, doing so bypasses the power of the Component Dispatcher. The best way to invoke Desktop components is using the RunComponentEx method of the Component Dispatcher library. The RunComponentEx method takes the name of the function to perform, such as DcProperties, and decides which actual COM component to launch. The context-based mapping of the function name to the component means that the same function may cause a different component to be displayed depending on the selected object's type, lifecycle state, user capability, or group membership. The rules governing which component to launch are managed using Documentum Application Builder.

The RunComponentEx method calls the IDcComponent::Init, IDcComponent::Run, and IDcComponent::DeInit methods of the component. Some components may perform an operation that results in the creation of new objects; for example, New, Copy and Check In. If the application requires the object IDs of the newly created objects in order to process them further, RunComponentEx can pass the object IDs of the new objects back to the calling code.

This sample code shows how to call RunComponentEx from Visual Basic.

```
' Declare the Variables
Dim ComponentDispatcher As DcComponentDispatcher
Dim objectitem As DCITEMSERVERLib.DcObjectItem
Dim reportMgr As DCREPORTSLib.DcReport

' Create an instance of the component dispatcher
Set ComponentDispatcher = New DCCMPDSPLib.DcComponentDispatcher
' Create an empty DcItems collection to hold the objects to
' display the properties of.
Dim NewItems As New DCITEMSERVERLib.DcItems
' Get a new ObjectItem and set its r_object_id
```

```
Set objectitem = New DCITEMSERVERLib.DcObjectItem
objectitem.ID = objID

' Add the ObjectItem to the DcItems collection after setting
' the item type Note: DC_OBJECT_ITEM_IID_STRING comes from a
' resource file supplied with the source
NewItems.Type = DC_OBJECT_ITEM_IID_STRING
NewItems.Add objectitem

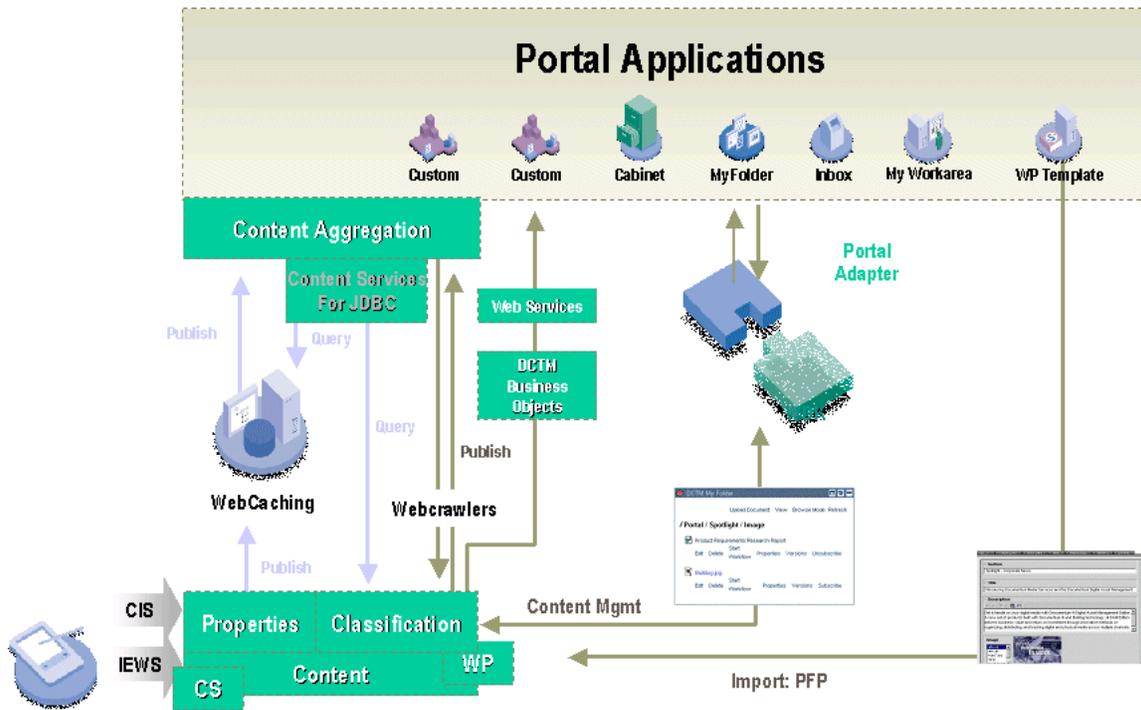
' Set up the values to pass into RunComponentEx.
invokerName = ""
docbaseName = dfSession.getDocbaseName
userName = dfSession.getLoginInfo.getUser
domainName = dfSession.getLoginInfo.getDomain

' Call RunComponentEx
Dim rv As Long
rv = ComponentDispatcher.RunComponentEx("DcProperties", _
                                         "DcDesktopClient", _
                                         docbaseName, _
                                         userName, _
                                         domainName, _
                                         NewItems, _
                                         frmMain.hWnd, _
                                         reportMgr, _
                                         "", _
                                         DC_NOTGLOBALCOMP)
```

Documentum Portal Integration Kit

Documentum Portal Integration Kit (PIK) extends the WDK framework by providing a set of Documentum components that expose common content management capabilities packaged as portlets (Java classes, JSP pages and XML definition files) that can be integrated into a portal. The PIK provides a “write once, run anywhere” approach where a Documentum component can be housed in many different portals and applications without code modification.

Figure 3-11. PIK Architecture



The PIK combines off-the-shelf, embeddable portal components with documentation and integration examples that enable portal vendors and customers to rapidly create adapters that bind the portlets to new portal environments. It consists of:

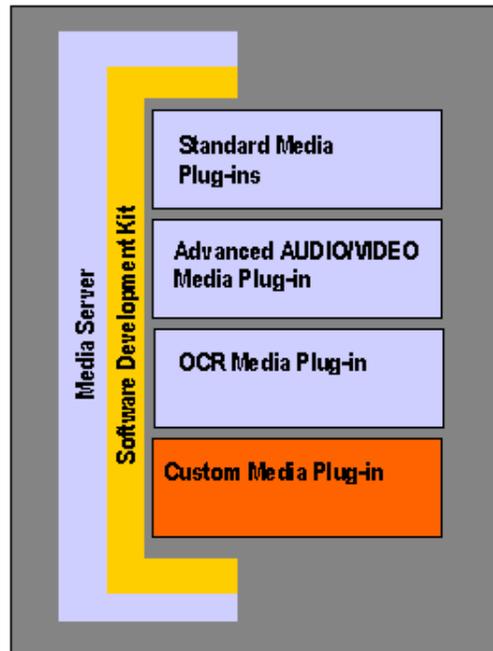
- Portlets for common activities, such as searching a repository, checking documents in and out, and viewing object properties
- Portal adapters for BEA WebLogic Portal, Epicentric, and Plumtree
- A portal adapter specification
- An API for building adapters for additional portal frameworks
- Adapter run-time support

The delivered portlets can be deployed as they are, or they can be configured to change their look and feel. For example, the presentation of an Inbox Component may be customized to add a Portal specific banner, or its behavior may be customized to add support for Portal specific caching. The PIK portal adapter kit also enables developers to adapt any WDK component as a portlet.

Documentum Media Services SDK

Documentum Media Services provides out-of-the-box capabilities for many common file formats. The Documentum Media Services SDK is a set of tools that allows developers to add media processing technology to Media Services by creating additional Media Plug-ins to handle proprietary or specialized file formats. The SDK consists of a Java API for connecting the Media Plug-ins into the Media Server, documentation for the APIs and for providing overall guidance, and sample source code.

Figure 3-12. Media Services Plug-Ins



Any program that transforms or analyzes content may be incorporated as a Media Plug-in. The programmer only needs to implement a simple interface and then configure Media Server to use the plug-in. The server looks after most of the housekeeping, such as running as a service, managing the request queue, fetching and storing content, managing the thread pool, and monitoring Media Plug-ins to ensure they behave well.

New plug-ins can be chained with out-of-the-box plug-ins to create sophisticated transformations. This also lets the developer leverage existing capabilities when developing a plug-in for a new file format. For example, the new plug-in might convert a proprietary file format into PDF. The developer could then leverage existing plug-ins to generate JPEG renditions for each page of the PDF.

To develop or extend a Media Plug-in, a developer does three things:

1. Write a plug-in that implements the Media Services Java interface
2. Write command files (which the server passes to the plug-in) for supported transformations and store them in the Documentum repository
3. Write or extend standard Transformation Profiles that define the file formats and parameters that apply for each transformation, and map these to the command files for the new plug-in

By following the Media Services design pattern and by performing these three steps, the new media plug-in capabilities will be available without user interface changes through any clients that use Media Services.

Deploying Content Applications

For mission-critical applications, most application development and testing should occur in a separate repository from the live, in-production repository. Developers and testers thoroughly test their new or updated applications on a test system, then deploy them to the production repositories. To support this model, Documentum Application Builder enables developers to package application elements into a single entity called a *DocApp*.

A *DocApp* is a virtual document that collects together all the elements of a custom application. It consists of some or all of these parts:

- Object types
- Document lifecycles
- Workflows
- Permission set templates
- Alias sets (used to assign symbolic names to users, permission sets, or repository locations)
- Executable elements
- Data dictionary information
- Repository objects, such as queries or folder hierarchies

For example, Documentum Desktop requires a *DocApp*, called *DcDesktopClient*. To use Desktop to access a Documentum repository, a customer installs *DcDesktopClient* in that repository.

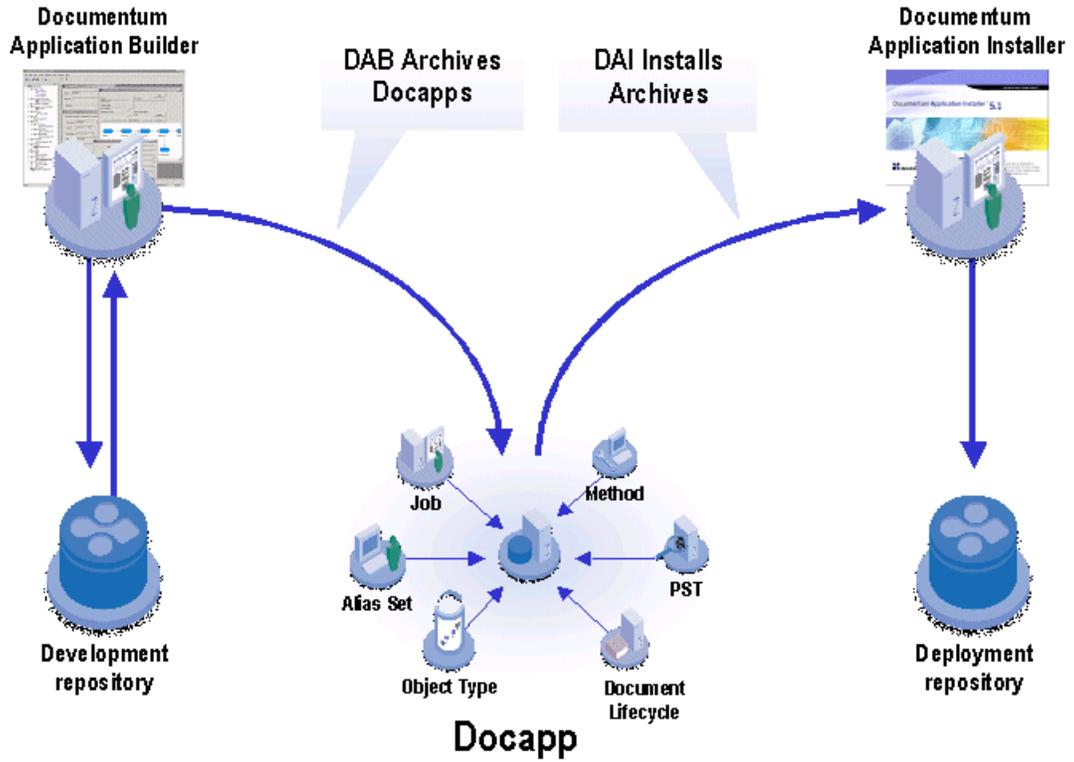
A *DocApp archive* is a portable representation of a *DocApp*, stored as a file on the file system. *DocApp* archives are created with Application Builder, then installed using Documentum *DocApp Installer*. The *DocApp* archive includes installation options for the objects contained in it, telling *DocApp Installer* whether to run a pre- or post-installation procedure, how to upgrade existing objects, which cabinet and folder objects are installed, and the location, permissions, and owners of objects in the target repository.

DocApp installer performs the following tasks:

- Detect and, if possible, resolve naming conflicts between the *DocApp* objects and the target repository
- Detect and, if possible, resolve object type conflicts between the *DocApp* and the target repository
- Unpack and install the contents of the *DocApp* archive

DocApp Installer performs these tasks automatically and writes a log of its actions in HTML format to a file that you specify. If conflicts arise, it either resolves them automatically or terminates the installation.

Figure 3-13. Deploying Content Applications as DocApps



Enterprise Platform Fundamentals

As a critical part of the IT infrastructure, Documentum demonstrates the key characteristics required in any production-ready enterprise platform. It is:

- Open
- Extensible
- Scalable
- Reliable
- Secure
- Portable
- Global
- Complete

Many of the features that embody these interrelated characteristics have been described earlier. This chapter provides an overview of how the Documentum platform fulfills these crucial requirements.

Open

Documentum 5 is thoroughly standards based, enabling it to integrate easily within existing IT infrastructures. The Documentum API provides standard APIs for WebDAV, FTP, ODBC, JDBC, and the Web services standards UDDI and WSDL. The Documentum platform is fully J2EE compliant (for Web-based applications) and supports the full range of Microsoft standards, such as .NET, COM, ASP, and Visual Basic. It provides out-of-the-box integration with enterprise applications and e-business platforms, including directory services using the LDAP standard. It has complete and configurable support for XML processing.

Extensible

The Documentum 5 platform provides an end-to-end solution for all content management needs — from creation/capture and management through delivery and archiving. However, the specific needs of each organization are unique, requiring extensions to the platform that embody the organization's business rules.

The object model that serves as the foundation of Documentum 5 is fully extensible, enabling customers to define custom object types that meet specific business requirements. The Business Objects Framework (BOF) provides a model for extending the range of available content management services. The platform supports the development of plug-ins in several key areas, such as user authentication, rich media handling, and legacy storage support. The open Documentum API ensures that customers can add content management capabilities into any application.

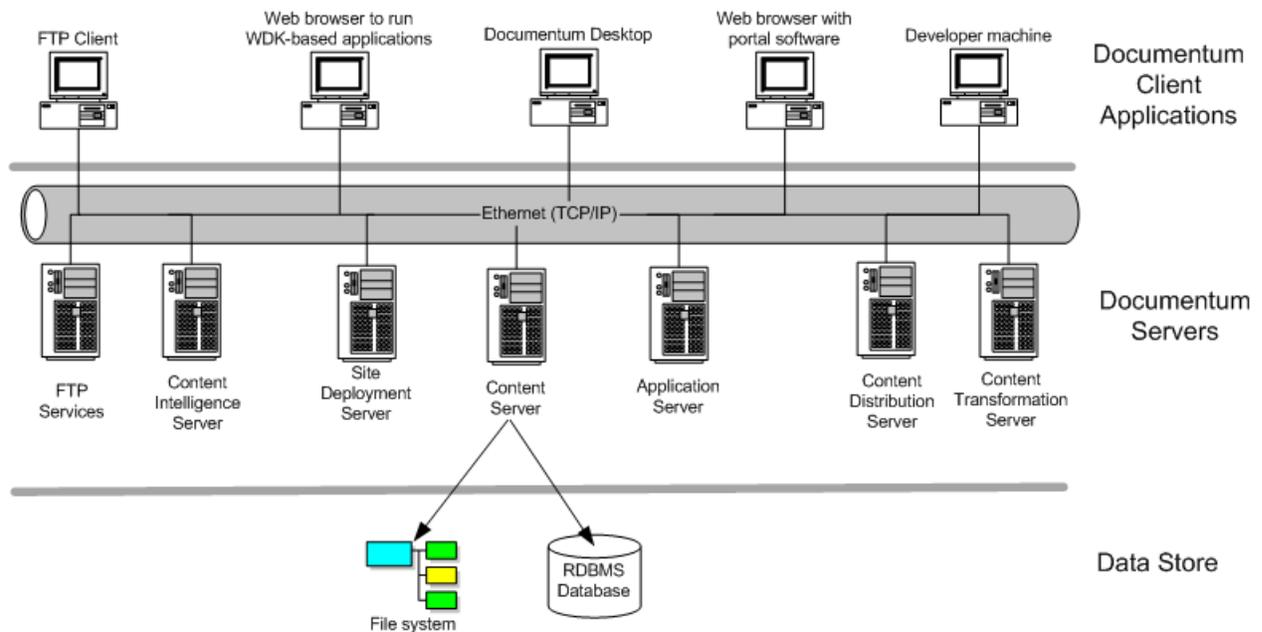
Scalable

Documentum 5 is designed for the global distributed enterprise. As a company's content management needs grow in size and complexity, the Documentum platform grows with it to efficiently manage ever-increasing volumes of content, high traffic loads, more users, and complex workflow processes, while maintaining high system performance. Documentum can support geographically distributed sites, handling the network latency and language issues that global enterprises face. It can handle these increasing loads in a cost-effective manner. The Documentum architecture has been designed to take full advantage of the scalability of the underlying hardware platform by utilizing multiprocessor systems as well as clustered environments.

Multitiered Deployment

An enterprise software implementation can be logically partitioned into layers of functionality, usually referred to as *tiers*. Each tier has responsibility for a distinct portion of the overall deployment, and within each tier, there can be one or more parts. In a typical content management implementation, the three main tiers are the client tier, the server tier and the data tier. (In an Internet-based environment, the application server is sometimes considered an additional tier that comes between the client and server tiers; alternatively the application server can be considered part of the client or server tier.)

Figure 4-1. Three-Tier Deployment



The client tier contains the software components dealing with user interfaces and user interaction. Client software displays content in a Web browser or standalone application and enables a user to view, edit, or otherwise manipulate the content.

The server tier makes content management services and capabilities available to clients. Servers communicate with clients using well-defined DFC application programming interfaces. The configuration of the servers on the server tier implements the customer's specific business rules and policies.

The data tier is where servers permanently store content, metadata about the content, the state of server objects and processing, and other information. Central to the data layer is one or more database servers that house the stored information.

The number of machines on each tier is extremely variable, depending on the volume of the content being managed and the complexity of the content applications. A key advantage of this distributed architecture is that new or more powerful machines can easily be added at whatever point is causing a bottleneck.

Horizontal and Vertical Scalability

The Documentum architecture supports both *horizontal scalability* and *vertical scalability*.

A software application scales horizontally (or "scales out") when it can be deployed not only on multiple servers across tiers, but also on multiple servers within a single tier. For example, multiple Content Servers can manage the same Documentum repository. Horizontal scaling implies that the load is balanced among the machines so that resource consumption between the servers is uniformly shared. This capacity growth strategy is popular because it allows a company to take advantage of the most competitively priced hardware, grow capacity incrementally, and provide cost-effective failover for high

availability. Each tier of the Documentum platform scales out across multiple servers, allowing customers to effectively build and grow highly-available, low-cost systems with excellent performance. Transparent load balancing at each tier helps to provide not only uniform resource consumption, but also continuous operation for high availability.

A software application scales vertically (or “scales up”) within a host when one or more tiers can be run within a single host and capacity can be increased by adding CPUs, memory, and disks. This type of scaling is popular in environments where maintenance costs need to be maintained. It can provide performance advantages as well. The Documentum platform takes full advantage of high-performance servers, facilitating data center consolidation and significantly reducing the cost of managing large-production content management environments. Documentum has a minimally synchronized, multi-threaded architecture, so that throughput is not limited by a single-threaded function or resource contention. The partitioning and load balancing used in the multi-server environment is also effective when scaling within a large symmetric multiprocessing (SMP) host.

The Documentum platform scales in all areas, including:

- **Content objects.** Content Server can handle repositories ranging from small departmental applications to enterprise implementations with thousands of users and billions of objects. A single Documentum repository is capable of handling 256 peta-objects (256×10^{15} objects), well above the threshold of even the largest enterprise content archive. Repositories can be combined into a federated environment for essentially unlimited capacity. Each object has a globally unique identifier, avoiding the bottlenecks that would occur with relational identifiers. Content Server has been verified to provide excellent response time with over 100 million objects. The Documentum platform also supports repository federations, which are groups of cooperating repositories that share common definitions to ensure the integrity of cross-repository operations.
- **Concurrent users.** To increase the number of concurrent contributors, each tier on the “authoring” side of the content management system must scale. The application server and HTTP server tier can be horizontally partitioned across multiple servers through the use of network load balancers. The Content Servers for a single repository can be partitioned across multiple servers in a similar fashion (an “active-active” server cluster). Content Server clients (Web-tier or Desktop) transparently load-balance their connections from a list of available Content Servers provided by the Documentum DocBroker.
- **Concurrent workflows.** As the number of content authors simultaneously promoting content through the various workflow stages increases, the ability of the workflow engine to scale increases in importance. A workflow typically consists of manual and automatic tasks. Individual contributors complete the manual tasks, and workflow agents execute the automatic tasks. Documentum supports multiple, parallel workflow agents to service the same automatic task queue, ensuring that automatic tasks are completed with minimal delay.
- **Content distribution.** Site Delivery Services and Content Distribution Services allow enterprises to rapidly and efficiently distribute content to multiple remote sites automatically, using centrally maintained source content.
- **Geographically dispersed users.** Scaling across an enterprise rarely involves a networking environment that is equal for all users. Distance can significantly

increase network latency for some users; other users may have limited access to bandwidth; yet other users might be disconnected for long periods of time. Documentum supports a rich array of distribution and replication features for content and metadata that are optimized for these real-world environments. On the server tier, the platform supports multiple character sets and locales, as well as distributed and federated repositories. With distributed content, the system provides proximity-based load balancing, ensuring that content is stored and retrieved from local repositories. On the client tier, Documentum provides persistent caching of metadata to minimize network traffic. It also supports an “offline” mode for client applications.

High Performance

As the number of users and volume of managed content expands, and the size and speed of the servers increases, the efficiency with which information is exchanged across the network becomes ever more critical. Improving the performance of servers cannot have the intended effect if the network becomes a bottleneck. Efficient use of the network can be especially important to global organizations, where network nodes may be spread across countries and have latencies of over 250 ms per message.

The biggest consumers of network resources in a multitiered system are:

- Establishing and maintaining connections between machines
- Transferring data (software objects and content)

Minimizing Connections

When a client application needs to interact with a server, it must first establish a *session* with the server through DFC. The session represents a connection through which the client can send instructions and requests to the server. The client maintains the session for as long as it needs the server connection, then disconnects. When creating a session, these steps occur:

- A network connection is established between the client (Web-based or Desktop) and the Content Server
- A thread or process is initialized on the Content Server to support the session
- A network connection is established between Content Server and the RDBMS
- The client user is authenticated by the Content Server

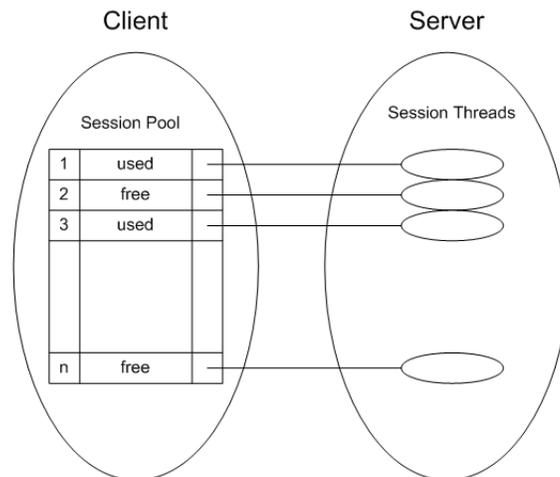
If a DFC-based client application needs to connect to a Documentum repository to run a simple query then disconnect, the process of connecting and disconnecting can take up to 90% of the total transaction time, with the query itself taking only 10%. With many clients repeatedly connecting and disconnecting from the server, it is easy to see how resources would be inefficiently consumed.

Because of the overhead associated with establishing a session, the most efficient strategy is to enable clients to *reuse* sessions that already exist. Rather than destroying the session object when a client is done using it, the client retains the session but marks it as idle;

when it needs another session, it reactivates the idle session rather than creating a new one. This approach is called *connection pooling* or *session pooling*.

A connection pool is a pre-established collection of sessions that can be shared between different client applications. Without connection pooling, when a session is disconnected, the session object is destroyed, the server process terminated, and the network connections dropped; in short, the work involved in establishing the connection is lost. With session pooling, the established session is retained and reused for a different application. Connection pooling is particularly powerful when the “client” is the application server running Documentum WDK-based content applications.

Figure 4-2. Connection Pooling



Connection pooling results in less overhead for establishing connections. It also results in significantly lowered resource requirements as sessions are shared amongst a number of users. Each request is handled by any available shared session from the pool.

Minimizing Data Transfer

Another potential drain on performance and network bandwidth is sending large volumes of data between clients and servers. The Documentum platform offers a number of strategies for reducing the amount of transferred data.

Some of these strategies relate to application design. One key principle is to request only the data necessary to complete a task. For example, to display a list of documents for a user to choose from, it is far more efficient to request just the documents' *attributes* from the server, not the documents themselves. Once the user selects a document, the application can retrieve the full document object for that single document.

Documentum Webtop provides an illustration of the principle. Because they are so intuitive, navigation trees are a common user interface element for locating documents in a folder hierarchy. In typical Web-based applications, the pages are rebuilt each time a user drills down into a cabinet or folder. Throughout the day, a user might navigate into 5 or 10 areas within the repository structure, causing those folder trees to remain expanded in the display. However, as each folder or cabinet is expanded, the size of the

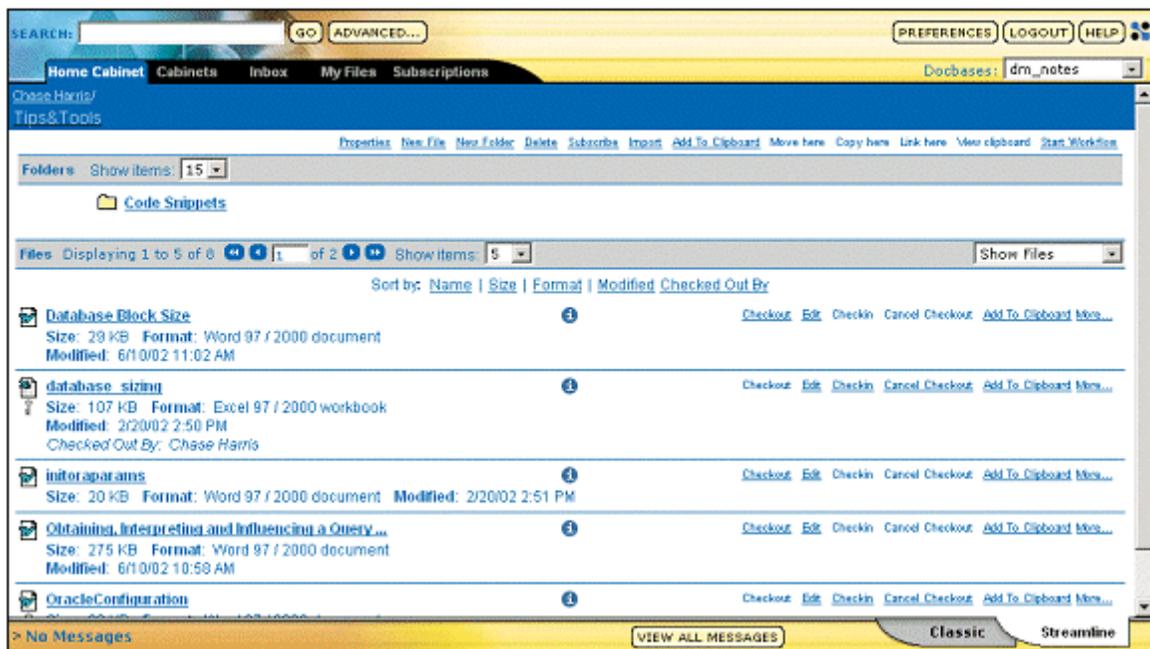
tree pane gets larger and larger, and response time gets slower and slower. The user must collapse folder paths in order to decrease that page size and get better response time. For users connected via high-speed networks, the cost of displaying the tree pane is not as significant as it is for users accessing the application via dialup or high latency connections.

For applications with navigation trees, the time to display the contents of a folder, Inbox, or search results is directly proportional to the number of items displayed in the tree. For example, when opening a folder with 500 documents, each document is processed in turn to build the appropriate links. This can produce a very large HTML page which would take a long time to transfer back to the browser.

Webtop offers two ways to handle this issue. First, it enables each user to set a maximum number of items to display at once. These settings can be changed dynamically based on the current requirements of the user. This functionality protects the user from the penalties of opening a very large folder, or issuing a very unselective search returning a large number of objects.

Second, Webtop offers *streamline view*, a user interface that does not use a navigation tree control. The streamline interface has been designed with the needs of remote users in mind. The lightweight interface has fewer graphics, requires fewer HTTP messages per page, and consumes less than half the bandwidth of “classic view” (with the navigation tree), making it ideal for users connecting to the repository from a dialup connection or with slower network connectivity. Webtop provides the ability to switch between classic view and streamline view on demand, to best suit the user’s connectivity and the task at hand.

Figure 4-3. Webtop Streamline View



Another way that Documentum applications minimize the exchange of data is through *caching*. The first time an application retrieves an object, it places a copy of that object in its local cache. The next time it needs to access the object, it can retrieve the local

copy. Instead of retrieving the object over the network again, the application exchanges a much smaller message with the server, verifying that its local copy is still the most recent version of the object. Documentum offers developers a choice of object coherency algorithms, enabling them to verify the local copy in the manner best suited to the environment. This persistent caching is particularly beneficial for infrequently changing objects.

Documentum's distributed content features also help minimize data transfer. With distributed content repositories, the system stores and retrieves content in local repositories, reducing the effect of slower network environments.

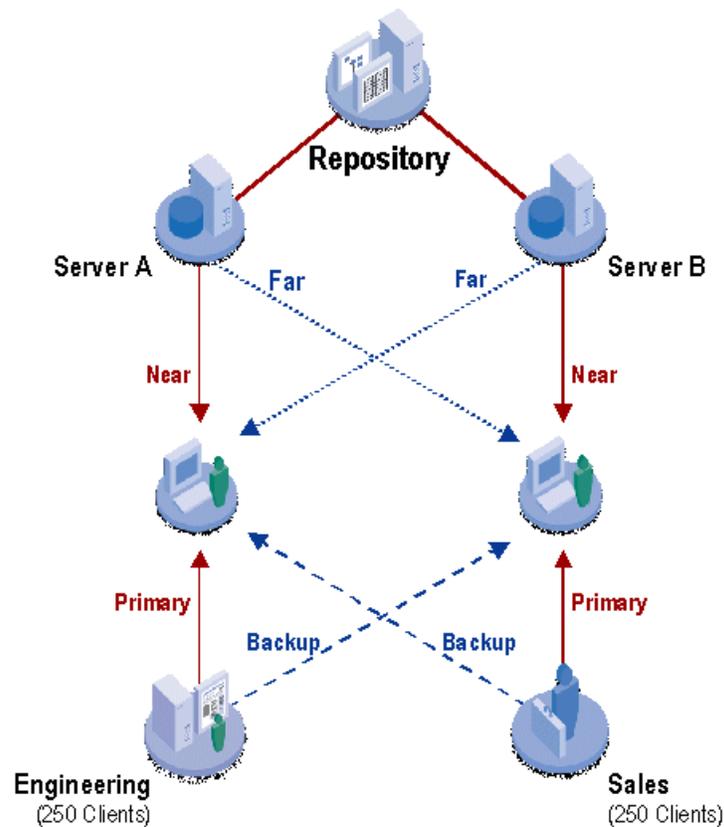
For activities that do not require immediate interaction with Content Server, Documentum client applications offer the option to "work offline." This feature enables users to download content from the Documentum repository to their local machine and work on them while disconnected from the repository. When they reconnect to the repository, the system synchronizes the offline content with the matching objects in the repository. The synchronization process updates the repository with any changes the user has stored on the local machine. Content can be correctly synchronized even if it has been moved or renamed in the repository. The user resolves any conflicts between the local copies and the repository during the synchronization.

Reliable

The Documentum 5 architecture follows a transactional model that ensures the integrity of the data in the repository at all times.

The multitiered architecture of the Documentum platform offers several options for ensuring that content management servers are available 24 hours a day. Multiple Content Servers can serve a single repository; if one of the servers goes down, the system automatically routes requests to another server for that repository. Repositories can be replicated, so that one serves as a backup for the other.

Figure 4-4. Content Server Clustering



Secure

Documentum 5 enables organizations to make their content available to a wider range of collaborators, but it does so without compromising the security of proprietary information. Content in the repository is completely secure: each piece of content has an associated security access control list that defines which users, groups, or roles can access it and which operations they can perform. For additional security, file stores containing content files can be encrypted. Administration files can also be encrypted, ensuring that sensitive information is not compromised.

Communication between the various components of the architecture is likewise secure. The channels between Content Server and other machines, including non-Documentum machines such as a directory server, can be handled through Secure Sockets Layer (SSL).

Web-based Documentum applications support single sign-on, where users are authenticated using a user name and password they have already entered for another applications. The authentication threshold is configurable through an extensible authentication plug-in framework, enabling Documentum applications to enforce two- or three-factor authentication using smart cards, certificates, or biometrics when this level of security is required.

As a further security measure, Documentum can keep an audit trail of all system activities, from user login times and content updates to changes in administrative settings. It supports the validation of digital signatures. The Documentum platform includes all of the features an organization needs to implement a system that is fully compliant with regulatory requirements, such as the 21 CFR Part 11 standard for pharmaceutical companies making submissions to the FDA.

Portable

Documentum 5 is supported on a variety of databases, operating systems and portal and application servers. Components of the system can be moved transparently from one supported configuration to another.

Table 4-1. Supported Configurations

Operating Systems <ul style="list-style-type: none">• Sun Solaris• Microsoft Windows• IBM AIX• HP-UX	Database Management Systems <ul style="list-style-type: none">• Oracle• Microsoft SQL Server• IBM DB/2• Sybase
Directory Servers <ul style="list-style-type: none">• Sun ONE Directory Server• Oracle Internet Directory• Active Directory	Application Servers <ul style="list-style-type: none">• J2EE• COM/.NET

Global

Businesses often have sites and customers around the world. Documentum 5 handles users and content from around the globe. It accommodates local languages, culture, and currency, including the storage of multilingual content in shared repositories. The Documentum platform can provide a single virtual repository spanning geographical boundaries. The virtual repository can take the form of a single distributed repository or repository federations, which are groups of cooperating repositories that share common definitions. The virtual repository gives users access to content regardless of the original language or geographical location.

Documentum 5 is Unicode compliant using the universal transformation format-8 (UTF8), which means that it can support single-byte languages such as English, French, or Italian, and double-byte characters like Korean or Japanese (Kanji). It supports clients in a variety of codepages and handles the transcoding between Unicode and other codepages.

Documentum provides localized user interfaces in seven languages: English, French, Italian, German, Spanish, Japanese, and Korean. Language support for these “Tier

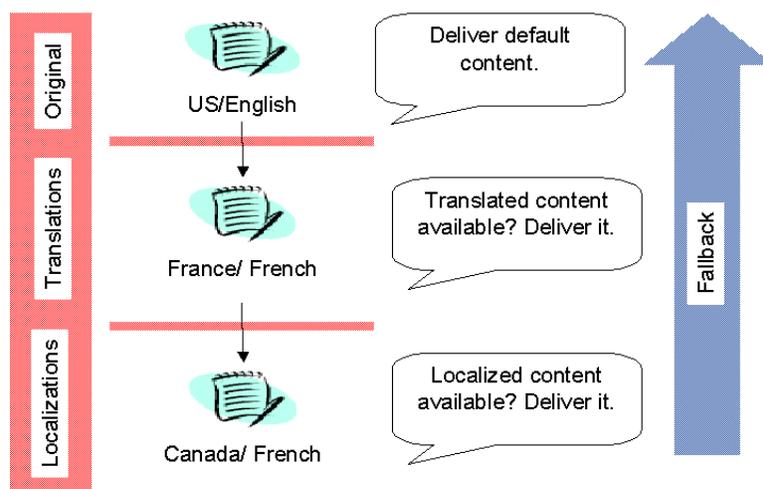
1" languages is delivered in the form of easily installed Language Packs, which are separately compiled and versioned add-on packages that provide a complete localized interface for a specific language. Documentum's Unicode support and Language Pack architecture give customers access to localized products built on the most recent base releases, and permit easy implementation of "Tier 2" localizations, simplifying localization and accelerating global deployment.

One of the key requirements of a globalized content management system is a repository that can store and manage multilingual content. Documentum's features and design do not make assumptions based on a single language or locale. Instead, Content Server can store metadata and content files from all languages and locales in a common repository. A unique feature is multi-language rendition management, which provides the ability to link versions of the same content in different languages. Users can access the content using a localized interface in any of the Tier 1 languages; users of Web-based applications can choose their preferred language when they log in. The full range of services is available for the multilingual content, including full-text searching in any of the Tier 1 languages as well as seven additional languages: Danish, Dutch, Finnish, Bokmal, Nynorsk, Portuguese, and Swedish.

Documentum provides built-in workflows to help automate the process of globalizing content. For example, Documentum Web Publisher includes workflows for exchanging content with translators. Through the use of Inter-Enterprise Workflow Services, external translators can participate as easily as in-house translators can.

Web Publisher enables contributors to deliver content in multiple languages and coordinates the publication of Web content in different language renditions. It supports fallback rules, which specify what happens when content is not available in the primary language for a Web site. For example, the rules for a French Canadian Web site might instruct the system to "fall back" to French (from France) if a specific localized file is not available, then to the United States English version if neither French version is available. Web Publisher includes built-in reports to track the status of the globalization progress.

Figure 4-5. Web Content Language Fallback Rules



Comprehensive

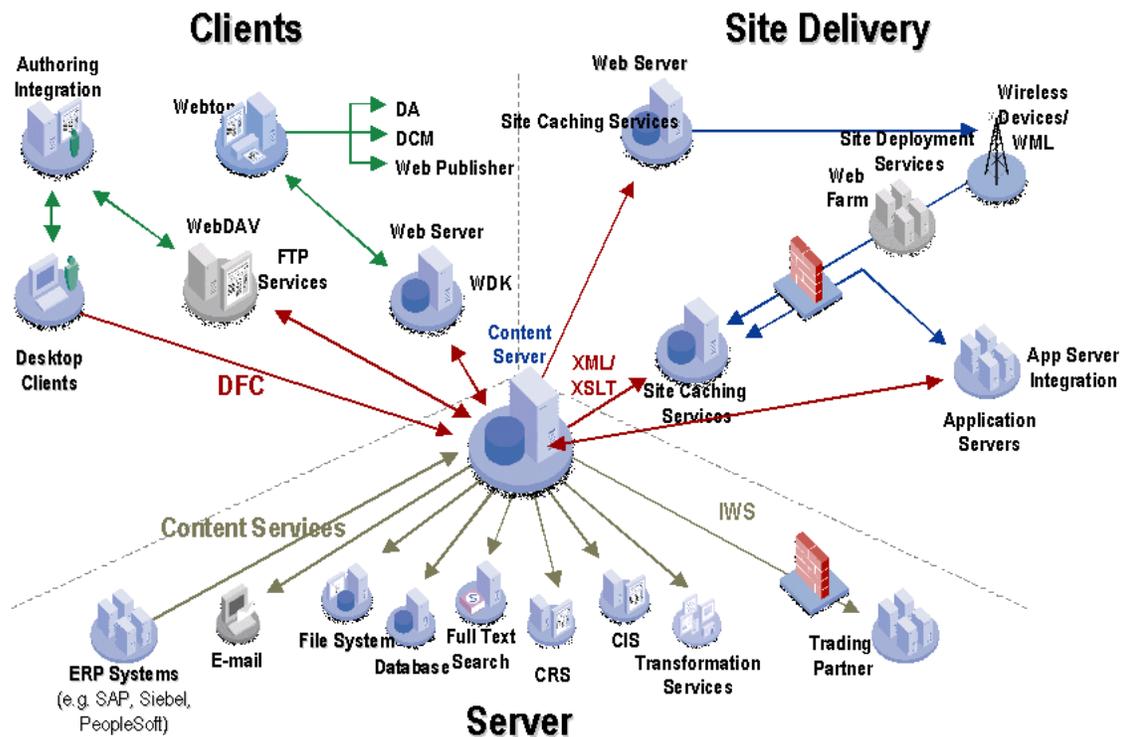
As the descriptions in this document demonstrate, Documentum 5 provides an end-to-end solution for all content management needs — from creation/capture and management through delivery and archiving. Documentum can manage content in any form. The platform is appropriate for traditional content applications as well as for business applications with more targeted content management needs. The open architecture enables customizations and plug-ins to extend the range of capabilities.

Documentum is recognized as the leader in enterprise content management by customers, analysts, partners, and developers. The Documentum enterprise content management platform has been field tested in thousands of the world's largest organizations and has already established itself as the platform on which customers and partners standardize their content infrastructure.

Documentum enables an entire ecosystem of partners, such as independent software vendors, systems integrators, hardware vendors, value added resellers, and original equipment manufacturers, which add value to the Documentum platform. For example, there are many commercial applications available from independent software vendors that leverage Documentum. They include accounts payable solutions that work in conjunction with ERP applications, contract management applications that augment supply chain management or CRM applications, publishing applications for use in regulated industries such as pharmaceuticals, and enterprise portal applications that deliver personalized information to a specific audience.

Every product innovation, every service enhancement, and every process improvement rests on an effective use of information. If you're not managing your content, you're not managing your business. Documentum can make your organization more agile, responsive, and productive. An investment in Documentum is an investment in the future of your business.

Figure 4-6. Documentum Universe



About Documentum

Documentum provides enterprise content management (ECM) solutions that enable organizations to unite teams, content, and associated business processes. Documentum's integrated set of content, compliance, and collaboration solutions support the way people work, from initial discussion and planning through design, production, marketing, sales, service, and corporate administration. With a single platform, Documentum enables people to collaboratively create, manage, deliver, and archive the content that drives business operations, from documents and discussions to e-mail, Web pages, records, and rich media. The Documentum platform makes it possible for companies to distribute all of this content in multiple languages, across internal and external systems, applications, and user communities. As a result, Documentum customers, which include thousands of the world's most successful organizations, harness corporate knowledge, accelerate time to market, increase customer satisfaction, enhance supply chain efficiencies, and reduce operating costs, improving their overall competitive advantage. For more information about Documentum, visit www.documentum.com or call 800.607.9546 (outside the U.S.: +1.925.600.6754).