

Database Publishing in Adobe FrameMaker 7.2

TABLE OF CONTENTS

- 1 Introduction
- 1 Applications of server-based publishing
- 10 The FrameMaker database publishing solution
- 17 Implementing a FrameMaker solution
- 23 Next steps
- 24 Appendix: Resources

Introduction

Database publishing is important to businesses of all sizes across all industries. For every enterprise, information must be distributed internally and to the world at large. From company phone directories to online product catalogs, database publishing enables organizations to communicate. A well-implemented database publishing system lets companies reduce cost, decrease publishing time, and increase information quality.

What is database publishing?

In general terms, database publishing is the automated transformation of source data into formats for presentation. In a typical database publishing application, content from multiple sources is aggregated and formatted for publishing to multiple formats, which can range from print and Adobe® PDF, to Hypertext Markup Language (HTML) on the World Wide Web, to Extensible Markup Language (XML) delivered to VoiceXML browsers and handheld devices.

Regardless of specific data sources, and particular format and delivery mechanisms, all forms of database publishing manifest some common and essential features: Data must be presented in a way so that users can easily find what they want; it must be formatted so presentation is effective; and presentation must often be customized for a particular target audience or even personalized for a single end user. The process of database publishing should be efficiently automated to meet the demand for timely delivery of consistently formatted information.



Overview

This whitepaper provides a general overview of database publishing and surveys implementations by industry and document type. It also discusses the unique capabilities that Adobe FrameMaker® offers, and describes some of the ways in which database publishing solutions can be implemented using FrameMaker, including integration with third-party database publishing tools such as PatternStream® software from Finite Matters Ltd. (FML) and Miramo® software from Datazone.

Applications of server-based publishing

Database publishing systems typically deliver content from web servers, application servers and database servers. A single system integrating multiple servers across a network or the Internet can deliver content to the entire enterprise or to the world at large. Server-based publishing solutions are typically tailored to meet the publishing needs specific to different industries, different document types and different data sources.

Server-based publishing can be considered from any of the following perspectives:

- By industry
- By document type
- By data source

Industry survey

While database publishing is important to most industries, the types of source data, required output formats, and desired levels of personalization and customization will undoubtedly be different depending on industry-specific needs and standards. To some companies, the most important documents are internally circulated; to others, publishing information to the world at large is either an important part of doing business or possibly, as with book publishers, the main focus of their business.

Server-based applications can be found in the following industries and organization types:

- Manufacturing
- Engineering
- Healthcare
- Pharmaceuticals
- Financial services
- Insurance
- Publishing
- Government

Manufacturing

A product catalog is critical to the success of most manufacturing companies. Product catalogs let prospective purchasers and distributors know what products are available, how much they cost, and other product details relevant to purchase and use of manufacturers' inventory. Such catalogs often contain many pages of complex tabular information, accompanied by extensive photographs or illustrations such as exploded parts diagrams. Manufacturing companies may also face significant technical documentation challenges for both internal information exchange and product documentation.

Since 1947

HYDRAULIC SUPPLY COMPANY STOCK PRODUCTS

Accumulators	Adapters	Air Brake Products	Air System Components	Control Cables	Cylinders
Flanges	Flexible Couplings	Flow Meters	Gauges	Hose Exchangers	High Force Hydraulic Hose & Fittings
Hose Assemblies	Hose Assembly Equipment	Hose Barbos	Hose Clamps	Hose Protection	Hose Reels
Hydrostatics	Lubricating Fluids	Motors	Performance Products	Pipe Fittings	Pipe Nipples
Pressure Washers	Pump Adaptors	Ramps	Quick Disconnects	Reels	Seats
Steering Units	Support Clamps	Swivel Joints	Thread Sealants	Tubing & Tube Fittings	Tube Bending
Valves	Wrenches				

Quick Reference Page

Company Profile	2
Table Of Contents	3
Catalog Guide & Sales Information	4
Product Index	13
Model Number Index	43
Technical Information	119
Catalog Part Number Index	1209

Tel: (800) 507-9651
Fax: (954) 845-9113
www.hydraulic-supply.com

Catalog SPC-01-2002 \$25.00

Directional Valves, Manually Operated

Hydraulic

2-Way 2-Position Pull-To-Open Manual Valves Vickers MPV1-10 Series To 3000 PSI & 12 GPM

Features: Cartridge style (Requires a machined cavity or housing for installation). Pull-to-open knob.

Application: For on-off control of a single stream of fluid. Pull-to-open. Release to close (spring returns valve to closed).

Maximum Operating Pressure: 3000 PSI.

Rated Flow: 12 GPM.

Temperature Range: -40°F to 248°F (Extreme Limits).

Recommended Filtration Level: ISO 4406 Cleanliness Level 18/16/13 or cleaner.

Cartridge: Model MPV1-10*, sold separately.

Housing: Cavity size C-10-D, sold separately.

2-Way 2-Position Pull-To-Open Manual Valves Vickers MPV1-10 Series To 3000 PSI & 12 GPM

Catalog Part No.	Vickers Part No.	Vickers Model	Housing	Seal Material	Wt.	Price Each
C1B734	365589	MPV1-10-R-D	None (Select from table below)	Buna-N	0.25	

Housings for Vickers MPV1-10 Manual Valve Cartridges						
Catalog Part No.	Part Number	Description	Wt.	Price Each		
C1B734	366191	Std C-10-2 Light Duty Aluminum Housing (3000 PSI Max) with SAE 8 Ports	0.38			
C1B734	366190	Std C-10-2 Light Duty Aluminum Housing (3000 PSI Max) with SAE 8 Ports	0.34			
C1B734	366201	Std C-10-2 Light Duty Aluminum Housing (3000 PSI Max) with 1/4 NPT Ports	0.38			
C1B734	366192	Std C-10-2 Light Duty Aluminum Housing (3000 PSI Max) with 3/8 NPT Ports	0.37			
C1B734	366189	Std C-10-2 Light Duty Aluminum Housing (3000 PSI Max) with 1/2 NPT Ports	0.34			

Detailed housing dimensions page 87

This page is part of a complete catalog which contains technical and safety data that must be reviewed when selecting a product.
1999-2001 Hydraulic Supply Co. 1-800-507-9651 www.hydraulic-supply.com

653

Product catalogs frequently contain complex illustrations and tabular information.

Information relevant to catalog production and technical documentation is generally housed in one or more databases. Related assets, such as graphics and word processing files, can be housed in the same or a different database, on file servers or in content management systems. Data and document sources may also be part of, or may interface with, an Enterprise Resource Planning system. Because product catalogs are time-sensitive, it is generally a requirement to automate the database publishing process.

Depending on the number, complexity, and volume of the products they manufacture, manufacturing companies can place great demands on a database publishing system.

Engineering

The engineering process is typically documentation-intensive. The documents required for publication of engineering information can be visually complex, including graphics such as flow charts, schematics, and Computer Aided Design (CAD) diagrams. For this reason, the publishing tools used by engineering companies and their departments must effectively integrate graphics and text.

Engineering is typically a collaborative effort, and organizations with an engineering focus will often house shared information in databases and file management systems. In mission-critical engineering projects, database publishing processes benefit from the highest degree of automation available.

Healthcare

Healthcare is an information-intensive industry. Patients need to know where they can go for care, the policies of their healthcare providers, and information specific to their health condition. Physicians need to know the network of physicians to whom they can refer patients, current research regarding therapy and medication, and information about their patients.

One way that patients find physicians is by accessing provider directories, which are typically published by health plans, clinics, or medical groups. Timeliness of directory information is important: Physicians frequently change locations, open or close their practices, or change status with a particular health plan or medical group. Automated database publishing is essential in meeting the demand for current, accurate information.

Provider directories typically integrate front matter sections that include information such as plan or provider policies, so publishing tools for directory production must be able to integrate word processing files with data-generated content. Provider directories are often formatted in a multiple-column layout and may have multiple indexes. With advances in the personalization of information, an increasing number of health plans and medical groups offer personalized provider directories (for example, custom directories of physicians within a selected radius of a patient's home, which are typically delivered physically as hard copy or electronically as PDF files).

Pharmaceuticals

One branch of the healthcare field with special database publishing needs is the pharmaceutical industry. This industry must inform physicians and other healthcare providers of current product availability, indications, dosages, currently known side effects, and potential drug interactions.

Pharmaceutical companies must maintain high standards for the accuracy and timeliness of the information they publish. First and foremost is the need of physicians and patients to access the most currently known information. The process of having new drugs reviewed and approved by legislative agencies is also important: Delay in approval due to late or incorrect information in the review process can cost patients access to treatment and the pharmaceutical company potential revenue.

Financial services

Financial services companies are faced with significant publishing requirements because of consumer demand and legislative mandate. Shareholders are typically entitled to periodic financial reports; government agencies place substantial demands for both reporting to the agencies themselves and for providing information to shareholders and the public at large. The formatting requirements for publication of financial information often include tabular data accompanied by graphs and/or charts.

The trend towards personalized content is perhaps strongest in the financial services arena. Consumers of financial information want to be able to access reports reflecting their current portfolio, based on the latest activity in their account and the latest valuation of their assets. To meet such a time-critical need, publishing applications must often access multiple systems in real time, and output finished print or electronic versions of documents in real time on demand.

Insurance

Insurance companies need to put great care into all of their documents—from insurance policies to the communication of rate and policy information to vendors, brokers, and the general public. Because much of the information that insurance companies publish represents a legal obligation, accuracy is essential.

Insurance policies and related documents are often managed with systems that merge data from back-end databases (for example, rate tables) into appropriate sections of the document, or component management systems that facilitate reuse of document components (for example, language common to many documents or document versions). Such documents typically must go through an extensive review and approval process. Database publishing systems for the insurance industry must have strong capabilities for dynamically merging data and document sources, formatting tabular information, and composing graphically rich documents that are easy to exchange, read, and understand. Distribution of documents and drafts—internally for review and to the public at large for final distribution—is often most efficient with PDF files.

Publishing

Although print output has always been essential, data-driven publishing has recently become an important dimension of the publishing industry. With increased demand for electronic delivery and an increasing number of required media and output formats, such as CD-ROM, HTML, and eBook, it has become desirable in many publishing contexts to maintain a distinction between the content and structure of source documents. Structured authoring with XML, or conversion of documents from an unstructured format into XML, will often maximize the potential for multichannel publishing.

Some publishers have made powerful innovations in the production of on-demand documents. On-demand publishing can be as simple as printing single editions or small quantities of a static book or publication in response to an order; it can also be a far more customized and data-intensive process. Recent innovations allow users to log on to a website, select sections from multiple publications (for example, selected chapters from different books) and integrate these sections into their own custom publication, which they can then receive in PDF, eBook, or printed form. In the publishing industry, it is becoming increasingly important to structure document content and fully automate the typesetting process.

Government

Governments represent a category of organization with unique publishing needs. Governments typically have vast amounts of information to manage and publish (for example, the United States is the world's largest publisher). Governments at several levels (federal, state, and local) need to publish to communicate with their citizens as well as internal and external customers. Data-generated document types published by governments include catalogs, directories, transit schedules, statistical reports, and budgets. Governments face constant pressure to become more efficient and make information readily accessible. Server-based publishing solutions can help meet these needs.

FY 2000 Recommended Budget Position Detail Report				
Countywide Positions				
Committee Name	FY 1999 Positions		FY 2000	
	Approved	Adjusted	Recommended	Amount Chg from FY 1999 Approved
Finance and Government Operations	1804.5	1933.0	1903.5	99.0
Public Safety and Justice	3677.5	3755.5	3743.5	66.0
Children and Families - Social Services Agency	2690.0	2649.0	2690.5	84.5
Santa Clara Valley Health and Hospital System	4837.5	4738.5	4927.0	289.5
Housing, Land Use, Environment and Transportation	719.0	740.5	744.5	25.5
Total Positions	13444.5	13816.5	14029.0	584.5

Finance and Government Operations Position Detail				
Budget Unit Name	FY 1999 Positions		FY 2000	
	Approved	Adjusted	Recommended	Amount Chg from FY 1999 Approved
0101 Supervisorial District 1	7.0	7.0	7.0	0.0
0102 Supervisorial District 2	7.0	7.0	7.0	0.0
0103 Supervisorial District 3	7.0	7.0	7.0	0.0
0104 Supervisorial District 4	7.0	7.0	7.0	0.0
0105 Supervisorial District 5	7.0	7.0	7.0	0.0
0106 Clerk Of The Board	30.0	31.0	34.0	4.0
0107 County Executive	68.0	74.0	69.0	1.0
0110 Assessor	272.0	278.0	272.0	0.0
0112 Measure B Trans Improvement Pgm	0.0	2.0	2.0	2.0
0120 County Counsel	96.5	104.5	106.5	10.0
0610 County Library	168.5	207.5	204.0	16.5
0130 GSA Intergovernmental Services	99.0	100.0	99.0	0.0
0140 Registrar Of Voters	33.0	34.0	34.0	1.0
0145 GSA Data Processing	176.0	190.0	204.0	28.0
0190 GSA Communications	115.0	116.0	115.0	0.0
0263 Facilities Department	211.0	263.0	239.0	24.0
0126 Personnel, Training, & Labor Relations	103.0	114.0	112.0	3.0
0132 Risk Management & Employee Benefits Services	67.0	68.0	66.5	0.5
0116 Controller/ Treasurer	78.0	82.0	81.0	3.0
0112 Tax Collector	66.0	66.0	63.0	-3.0
0114 County Recorder	65.5	67.5	69.5	4.0
0118 Purchasing	57.0	59.0	57.5	0.5
0148 Department Of Revenue	69.5	69.5	70.0	0.5
Total Positions	1804.5	1933.0	1903.5	99.0

Public Safety and Justice Position Detail				
Budget Unit Name	FY 1999 Positions		FY 2000	
	Approved	Adjusted	Recommended	Amount Chg from FY 1999 Approved
0200 District Attorney Family Support	335.5	347.5	345.0	9.5
0202 District Attorney Administration	420.0	431.0	442.0	13.0
0303 District Attorney Crime Laboratory	42.0	45.0	46.0	4.0
0204 Public Defender	215.0	217.0	216.5	1.5
0215 Office Of Probation Services	36.5	36.5	36.0	0.5
0230 Sheriff Services	575.0	578.0	585.0	10.0
0231 County/City Operations	112.0	112.0	117.0	6.0
0235 DOC Contract	714.5	714.5	713.5	-1.0
0240 Department Of Correction	394.5	405.5	397.5	3.0
0246 Probation Department	794.5	826.5	818.0	23.5
0293 Medical Examiner - Coroner	23.0	23.0	23.0	0.0
Total Positions	3677.5	3755.5	3743.5	66.0

Children and Families - Social Services Agency Position Detail				
Budget Unit Name	FY 1999 Positions		FY 2000	
	Approved	Adjusted	Recommended	Amount Chg from FY 1999 Approved
0501 Social Services Administration	2690.0	2649.0	2690.5	84.5
0509 GSA Nutrition Services To The Aged	6.0	6.0	6.0	0.0
Total Positions	2696.0	2649.0	2690.5	84.5

Santa Clara Valley Health and Hospital System Position Detail				
Budget Unit Name	FY 1999 Positions		FY 2000	
	Approved	Adjusted	Recommended	Amount Chg from FY 1999 Approved
0410 Public Health	324.5	634.0	615.0	43.0
0412 Mental Health	381.0	387.5	386.5	5.5
0414 Children's Shelter & Custody Health Services	270.5	270.5	288.5	18.0
0417 Bureau Of Drug And Alcohol Programs	143.5	155.5	162.5	19.0
0725 Valley Health Plan	28.0	28.0	29.0	5.0
0261 Valley Medical Center	3340.0	3271.0	3429.0	199.0
Total Positions	4837.5	4738.5	4927.0	289.5

Document types

Just as different industries have different database publishing needs, varied document types present varied publishing challenges. Documents can be either text-centric or data-centric; they can be technical, personalized, or both. Document size can range from single-page forms, to hundreds or even thousands of pages in the case of directories and catalogs. Documents of similar size often face similar data generation and page composition issues.

In the case of long, data-generated documents, sorting and grouping can be very important. An automobile parts catalog, for example, may be sorted by make, model, year, and subsequent part information such as part name or part number. Directories often have very similar sorting and grouping. In such cases, navigation can be aided by means of bleed tabs (that is, reverse-type vertical or horizontal bars that bleed to the side of the printed page, enabling easy section identification while thumbing through the catalog or directory), and running or “continued” headers indicating the context of particular pages within the overall grouping or sort. The indexes of such long, data-generated documents may also be grouped and sorted (for example, multiple indexes, each with a particular sort order, are often required).

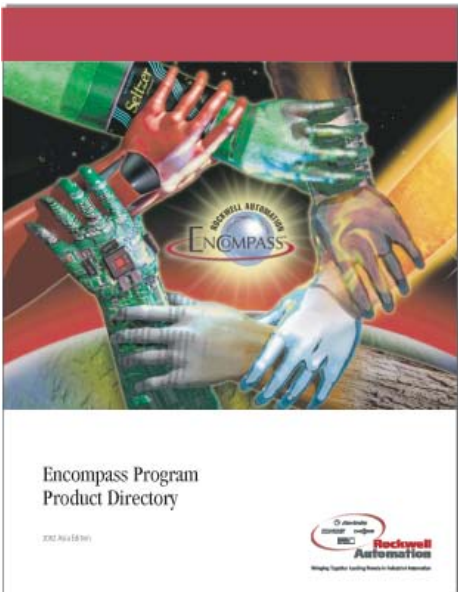
Document types that can be generated by Server-based applications include:

- Product catalogs
- Directories
- Invoices
- Financial reports
- Books
- Technical documentation
- Tabular information
- Real-time personalized documents

Product catalogs

Product catalogs are essential to those companies that need to make their products known to consumers and distributors. Product catalogs typically include tabular information and often contain images (either photographs or illustrations) representing some or all products. Automated tables of contents (TOCs), indexes, or cross-references are frequently a functional requirement for product catalogs as there are multiple ways a product can be referenced (for example, part number, name, or function). Products may have relationships that are important for the user (for example, one part may contain component parts or be compatible with specific other parts). Indexes, TOCs, and cross-references help the user effectively navigate the catalog.

Product information is typically housed in a database, but the production of catalogs can also include graphic and text assets from other sources. Database publishing systems suitable for product catalog generation must have the capacity to handle long documents with complex tables and a large number of graphic objects. The ability to import from word processing and other document formats (such as XML) is often important.



Product catalogs often integrate graphic assets, including photographs.

Directories

Directories are essential documents for people, companies, and industries. From phone books to internal staff directories and published association listings, directories provide a means for people to quickly and efficiently find the resources they need. While the key functionality of a directory is typically focused around some common core fields (such as company or person name, address, and phone number), specific directories may include a wide range of additional information. Healthcare provider directories, for example, may include a physician's specialty, provider type, hospital affiliations, foreign languages spoken, and indications of health plan membership. Directories may also include illustrations or photographs, though directories tend to be less graphically intensive than product catalogs.

For directories of substantive size, source information is housed in a database. To efficiently use the printed page, directories may be formatted in multiple columns. Directories typically include front matter (that is, text that provides information related to directory content); for commercial directories, this may include advertising. Front matter may come from word processing or page layout files. Directories can be quite large (for example, phone books), and efficient layout and ease of navigation are essential in these document types.

Directories are often formatted in multiple columns to efficiently use the printed page.

Invoices

Invoices are important to all businesses as they represent formal request for payment. In many cases, an invoice is a single page with few line items; however, it is quite possible and common for some industries to span invoices over many pages. In extreme cases, layout requirements for such long invoices can be similar to those of catalogs or directories.

More commonly, the scalability challenges faced by those producing invoices are due to the need to produce invoices in high volume. A company that must bill many customers on a monthly or other periodic basis, for example, faces the challenge of achieving high throughput while maintaining accuracy of information and consistency of layout.

Financial reports

Financial reports include documents such as prospectuses, financial statements, annual reports, and proxy statements. Financial reporting often has to include both high level summary information as well as detailed historical data. With financial reports, a picture often paints a thousand words: Charts, graphs, and other visual displays of quantitative information let end users see the characteristics of data faster and with far less effort than reading through unformatted data.

Financial information is data-centric and most financial reports come directly from a database. As the presentation of such information is often important to the corporate image of the financial institution or business presenting the report, design quality is also meaningful. As with invoices, the scalability challenges for financial reporting are most often challenges of high throughput rather than high page count or large document size. Timely distribution of accurate, well-presented information is critical to financial reporting.

Books

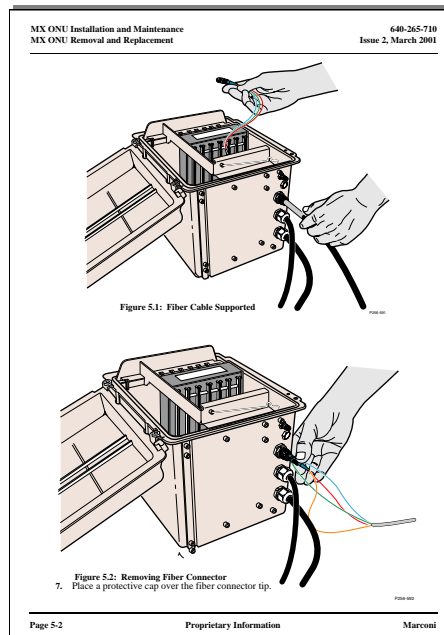
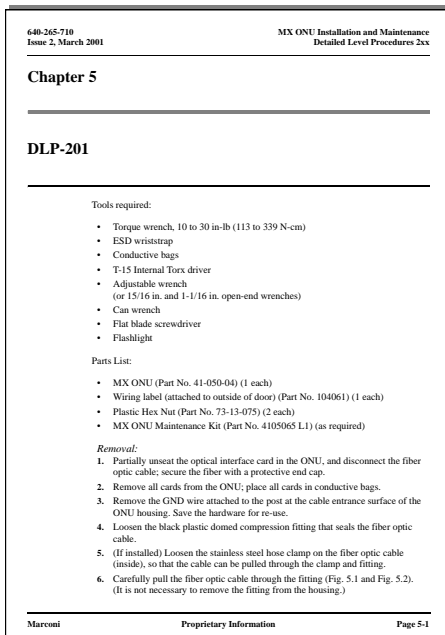
Of all the many document types, books tend to take the longest time to produce; Once produced, they tend to have a longer shelf life than most data-generated information. Books are not typically maintained in databases, though the movement towards structured authoring has led to a steady increase in the number of books maintained in a rigorously defined document structure.

Although books may not be data generated, the same benefits of automation available for information published from databases is possible for those books that are maintained in structured formats such as XML. When book content is available as XML, it is possible to repurpose such content for integration into “custom books” or output to new media formats such as eBooks.

The concept of a book as a paper-only document is steadily being eroded by advances in publishing technology. Adobe PDF already enjoys widespread use; it is often desirable to make chapters or entire books available PDF files. In extreme cases, books have been published exclusively as PDF files, eBooks, or both.

Technical documentation

Technical documentation is often mission-critical. Documentation of procedures (for example, airplane repair) demands the highest standards of clarity and accuracy. In addition to the integrity of data, text, and graphic objects used in technical documentation, effectiveness of formatting and page layout can be crucial in maximizing the clarity and usability of such critical information.



Technical documentation often integrates images and text to clearly document procedures.

While technical documentation is essential to engineering processes and product documentation, the documents and data associated with technical documentation quite often find their way into many aspects of organizational activity. The same CAD illustration, for example, may be used in the initial design of a product as well as in subsequent sales and marketing efforts such as brochures, catalogs, and advertisements. Data, documents, and graphics created or accessed in the course of technical documentation are often shared across departments.

Tabular information

Data-generated information is sometimes most effectively presented in the form of tables. In their simplest form, tables have a header row (identifying the content of the table's columns) and one or several other rows (each having multiple cells with one cell for each column). However, there are many ways that tables can be made more complex: A cell can span multiple rows or columns; tables can have column headers in addition to or instead of row headers (for example, a multiplication table); and complex formatting, even embedded objects including tables themselves, can exist within cells.

With long, data-generated documents such as directories and catalogs, it is not uncommon for a single table to span multiple pages—sometimes even the length of the entire document. In such cases, it is generally desirable to repeat row or column header information to provide a point of reference for each page of the document.

Real-time personalized documents

With the advent of the World Wide Web, there is increased demand for instant information tailored to the unique needs of the individual end user. For example, a user may wish to see their current stock portfolio, a listing of stores in a radius around their home address, or the results of a query based on criteria entered on a website. With automated database publishing processes, such instantly accessed information can be formatted for high quality presentation.

Data sources for server-side publishing

In much the way that the number of output media and formats has increased, the number of potential data sources available for use in database publishing applications has tended to increase over time. It is quite common for a single database publishing system to aggregate data from several systems—often in multiple formats—to produce output that appears to the user as if it came from a single source.

- Server-based applications can publish from data sources including:
 - Relational databases
 - XML repositories
 - Mainframe and legacy systems
 - Internet applications

Relational databases

The most common form of database is the relational database. Such databases are the traditional backbone of database publishing systems.

Relational databases store information in multiple tables, typically on database servers. The capacity of databases to store information continues to increase; even personal databases are now able to handle millions of records.

Relational databases are typically accessed by publishing applications via queries in Structured Query Language (SQL). An SQL query returns records from the database that meet a specific criteria (for example, the banking transactions for a particular customer for a particular time period). Records can be filtered to include specific fields and can be sorted based on any number of key fields. Database publishing applications typically process data received from such queries into formatted output.

XML repositories

With a small number of files, XML can be effectively stored as simple text files on a personal computer or file server. As the number of files or the number of participants in collaboration on a single file increases, however, it can become desirable to house XML in a system that can provide features such as version control, check-in/check-out and component reuse.

While it is possible to persist XML in relational databases by means of mapping the XML structure to that of a relational database, such mapping can, in many cases, be quite inefficient. For this reason, XML repositories often use other methods of data storage. Entire XML files or XML fragments can be retrieved by database publishing systems for formatting.

Publishing systems that can import XML and stylize it directly are generally more powerful at presenting such data; however, it is also possible for intermediate processing to transform XML into formats that even database publishing applications unaware of XML can understand.

Mainframe and legacy systems

While relational databases and XML repositories provide the typical data storage mechanisms used by most newly developed systems, there are still mainframe systems being produced with alternative means of data storage. Nevertheless, there are many legacy systems in organizations that are still doing the job of persisting data, perhaps with older storage methods, with effective reliability and speed.

In many cases, advances in enterprise application integration have preserved the life of legacy systems that had appeared slated for extinction. New applications built to interface with legacy systems can sometimes provide adequately effective access to the information housed in legacy databases. Database publishing applications can leverage such interfaces to provide access to legacy and mainframe data, presenting it with the same formatting characteristics as data from other sources.

Internet applications

Not all of the content flowing into database publishing applications comes from a persistent data store such as a relational database, XML repository, or mainframe system. It is now possible for data to come from Internet applications such as web servers, application servers or even web browsers. For example, a user requesting an online financial report will typically enter information into an HTML form. Upon submission of the form, they will receive a formatted report. The report may include information from the form as well as information accessed by querying a database.

With the rise of web services, an increasing number of systems exchange information by means of XML messages. There are, for example, standards for XML purchase orders and ordering processes that facilitate business-to-business transactions. Such XML messages can also serve as data sources for publishing systems.

The FrameMaker database publishing solution

FrameMaker offers a proven, powerful formatting engine that can play a central role in server-based publishing applications. FrameMaker meets the publishing needs of a wide range of industries, processing content from an array of data sources and file formats into formatted output reflecting a wide range of document types.

Core features of FrameMaker

At the core of FrameMaker is its robust page layout and formatting capability. This power can be used and extended in several different ways depending on specific needs: on both the desktop and the server (with the FrameMaker Server product); for authoring and publishing either unstructured or structured documents; and customized using the Frame® Developer's Kit (FDK).

Page layout with the FrameMaker document model

Much of the power of the formatting capability of FrameMaker comes from its rich document model. This document object model is accessible through its user-friendly Graphical User Interface (GUI) and can be controlled programmatically via the FDK.

The methods of defining document structure and page layout in FrameMaker are as efficient as they are powerful, allowing for reuse of styles and formatting at a number of levels: from defined character styles that can be applied to single words to master pages that control the layout of associated pages of a FrameMaker document.

Desktop and server capabilities

While the standard (desktop) version of FrameMaker offers complete desktop authoring and publishing capabilities, server-side publishing is possible with the FrameMaker Server product. The two products use the same publishing engine, allowing for a workflow in which templates and formatting rules created in the user-friendly desktop authoring environment can be applied directly to data streams from a wide variety of sources (including content authored in FrameMaker) in an automated, high-volume, high-throughput, server-based publishing solution.

Integrated structured authoring and composition

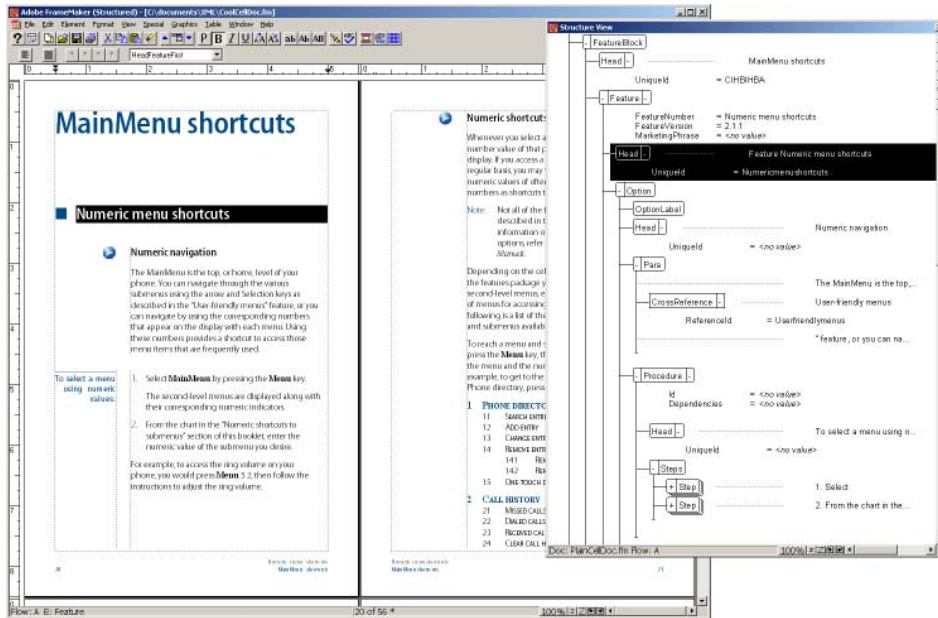
FrameMaker users are given a choice of two editing modes: structured or unstructured. This preference can be selected at program startup or through the Preferences dialog box. While unstructured mode is useful for relatively simple documents, structured authoring provides greater power for more highly organized documents or for cases where it is critical to assure a consistent structure such as a collaborative workflow.

The structured mode in FrameMaker provides XML support, letting you edit in conformance to an XML Document Type Definition (DTD). Read-write rules let you map XML elements to common document objects such as graphics, tables, and footnotes.

The Structure View enables navigation or section rearrangement, while the Element Catalog lets you see which elements are valid at a particular location in the document. FrameMaker guides conformance to the DTD throughout the authoring process, providing warnings of violations of the defined structure. FrameMaker offers complete validation functionality in addition to real-world flexibility, allowing users to save "broken" documents (that is, documents no longer conforming to the structure defined by the DTD) and enter corrections at a later time.

In addition to its XML authoring capability, FrameMaker lets you associate XML context with formatting rules. This feature enables the user to control precisely how the XML will look when printed or output to PDF. A FrameMaker Element Definition Document (EDD) is used to define context-aware mapping between elements and formatting. Such mapping allows FrameMaker to be used as a formatting engine for XML documents and data sources.

By combining integrated XML editing and formatting capabilities, FrameMaker provides the best of both worlds: simultaneous structured and “What You See Is What You Get” (WYSIWYG) authoring of the same document. While structure is important, a WYSIWYG environment offers productivity gains and eases the migration to structured authoring for those accustomed to working with word processing software.



FrameMaker provides simultaneous WYSIWYG and Structure View XML authoring capabilities.

Robust API to enhance customization

If your needs go beyond the functionality that FrameMaker offers out of the box, it is possible to customize the program using the freely available FDK. The FDK lets you control the FrameMaker application through its powerful Application Programming Interface (API). The API exposes all FrameMaker document objects to programmatic control, letting you write C language programs that can do anything an interactive user can do—and more.

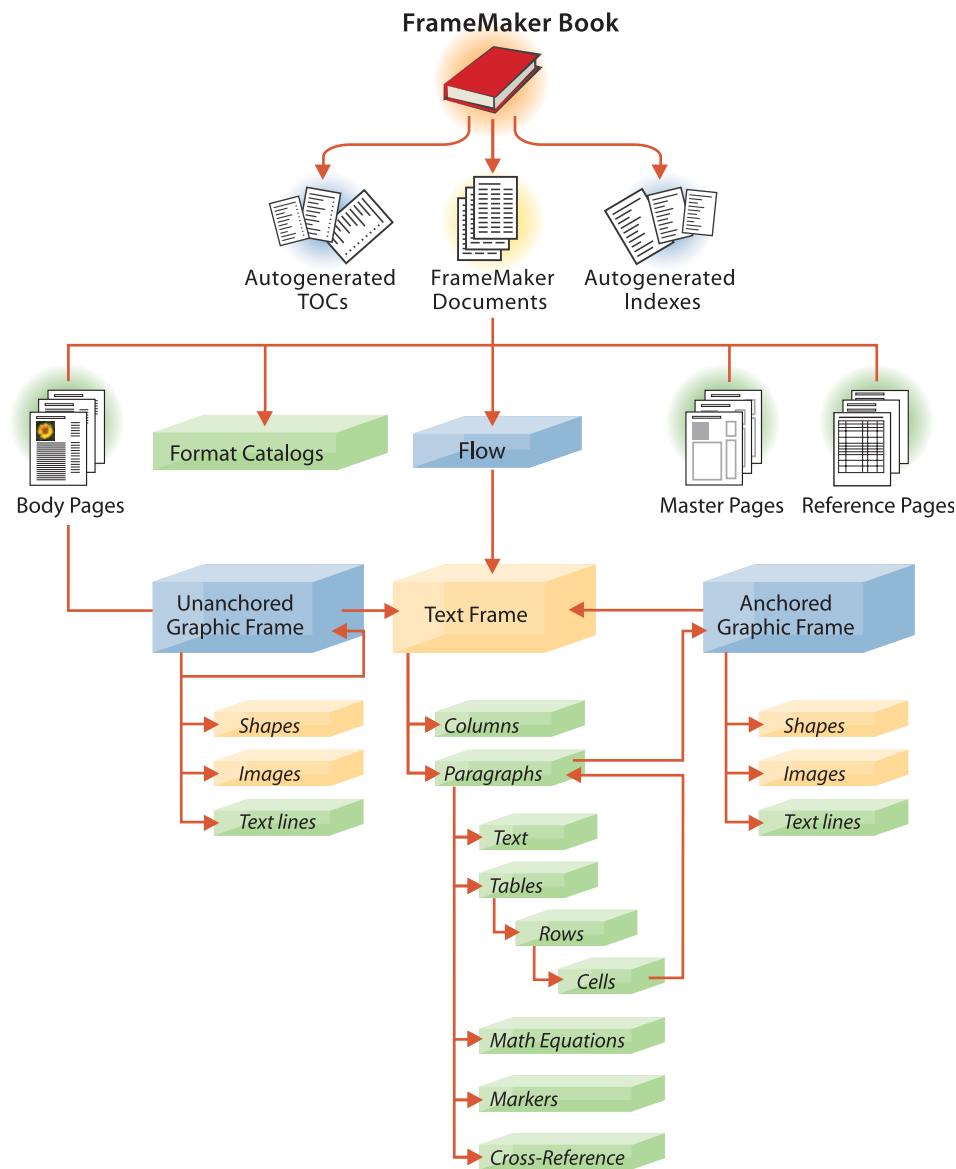
The FDK lets you create plug-ins that extend the capabilities of FrameMaker (for example, adding custom menu items that will run functions that you have created). Because of the extensibility provided by the FDK, FrameMaker enjoys integration with many third-party applications.

The FrameMaker document object model

FrameMaker provides a complete range of advanced publishing features, and its document object model includes many document objects: from pages to every sort of object found on a page (for example, paragraphs, tables and graphic objects) to objects used in the course of page layout (for example, paragraph and character formats, cross-references, and variables).

FrameMaker document objects

At a high level in the hierarchy of its document objects, the FrameMaker document consists of one or more *body pages*, which are visible when a document is printed or output as PDF files. In addition to body pages, there are two types of pages that are not visible in the printed output but function behind the scenes: *master pages* and *reference pages*.



FrameMaker offers a powerful document object model to meet a wide range of publishing needs.

Master pages provide general layout characteristics, their own static text and graphics, or both; body pages inherit layout, static text, and graphic objects from applied master pages. Reference pages serve as kind of a catalog of objects; both body pages and master pages can reuse objects from reference pages such as commonly used graphics. Specialized reference pages can also contain hypertext commands, formatting information, and mappings for converting to XML and HTML.

On body pages, text and graphics appear in rectangular frames. *Text frames* control placement of the document text and *graphic frames* control the placement of graphics. Graphic frames can be either anchored (that is, they stay with the text to which they are “anchored” and move as you make text changes) or unanchored (that is, they assume an absolute position on the page). A unique kind of unanchored graphic frame is the page itself. (Internally, FrameMaker considers pages “page frames”—a special class of unanchored frame.)

FrameMaker documents also contain *format catalogs*, consisting of reusable formats such as paragraph and character styles. The *flow* objects provide a means of controlling how text in that particular stream “flows.” Text frames and paragraphs can be associated with a particular flow to produce, for example, a multiple-column layout found in newspapers and magazines.

There are many ways that FrameMaker document objects can contain one another, including recursive models in which some types of objects can directly or indirectly contain objects of the same type. While such object relationships can seem complex when analyzed in detail, FrameMaker provides a user-friendly interface that makes the development of complex document templates a straightforward and intuitive process.

The power of MIF

Any FrameMaker document can be saved in Maker Interchange Format (MIF). This tagged text format represents a serialization of its document object model: all of the document’s objects and the relationships between them are fully described in the human-readable and text-based format created when you save your FrameMaker document as MIF. Documents can be saved as MIF and loaded back into FrameMaker without losing any information.

MIF offers many interesting possibilities for the architecture of publishing applications. Sun Microsystems, for example, has developed a Java program called the MIF Doclet, which automatically generates documentation of Java programs from declarations and comments in the source code itself. Information about the MIF Doclet is available at <http://java.sun.com/j2se/javadoc/mifdoclet/index.html>.

For database publishing, it is possible to write applications that generate MIF output based on iterating through records in the database and wrapping fields from the data in appropriate MIF tags. MIF can also be produced as the result of Extensible Stylesheet Language Transformations (XSLT) of XML. Resulting MIF from such processes can be loaded into the desktop version of FrameMaker, and printed or output as PDF or other formats. It can also be streamed into FrameMaker Server and published as a completely automated process.

Graphic objects and graphic frames

In the FrameMaker document object model, the meaning of *graphic object* is inclusive of objects that might not seem very “graphic” (for example, text frames). Graphic objects include:

- Anchored graphic frames (containers for graphic objects “anchored” to a specific location in the text)
- Unanchored graphic frames (containers for graphic objects not associated with a specific location in the text)
- Simple geometrical shapes (for example, lines, arcs and polygons)
- Text lines (single lines of text typically associated with callouts or graphic captions)
- Text frames (containers for text in a flow)
- Imported graphics in a wide variety of formats including bitmap, Scalable Vector Graphics (SVG), PDF and Encapsulated PostScript® (EPS), as well as native Adobe Illustrator® and Adobe Photoshop® files
- Math equations

Anchored graphic frames add a powerful dimension to page layout capability. With anchored graphic frames, templates can be built that allow graphics to keep next to relevant text, even if the content varies considerably between documents and the text and graphics reflow between pages.

Master pages and body pages

Master pages define a general structure for page layout that can be reused by the body pages of a document (that is, every body page of a document inherits the characteristics of an applied master page). A double-sided document contains at least two master pages (for left and right pages, respectively) and a single-sided document uses only the right master page. Custom master pages can be created for special types of pages; page-specific layouts can also be created directly on body pages themselves.

You can draw or import graphics anywhere on a master page. These graphics will then become the background of associated body pages. Master pages can contain two different types of text frames, alone or in combination:

- One with a named and tagged text flow, so you can type document text into these frames in associated body pages.
- One that contains an untagged (and therefore unnamed) text flow.

The latter type appears in the background on associated body pages and is generally used for headers and footers; the text can only be edited on the master page itself.

Master pages aid in keeping content distinct from presentation: With such a workflow, you can update the design of your published materials independently of your data source. In FrameMaker, a change to a single master page can dramatically change the appearance of an entire document.

Paragraph and character formatting

Each FrameMaker document contains catalogs of paragraph and character formats. Like master pages, these formats are key elements in the template-driven workflow of FrameMaker. By using the format catalogs, you can ensure document style consistency. Paragraph and character styles can also be used to define XML styles and assist in export to formats such as HTML.

Paragraph formats affect entire paragraphs; character formats affect any range of selected text (typically, phrases or individual words within a paragraph). The Paragraph Designer enables you to define and save detailed formatting information for entire paragraphs (for example, default font, space above and below, “keep” options, widow/orphan control, and autonumbering). The Character Designer offers a more limited palette of options (for example, font family, size, and font weight).

FrameMaker tables

One of the most powerful features of FrameMaker for database publishing is its ability to handle tables. FrameMaker tables can span many pages and include complex structure with detailed formatting. Such capability can be essential to documents such as parts catalogs and technical documentation.

A table typically contains a title, a header row, one or more body rows, and possibly a footer row. Using the Table Designer, you can define default cell margins, spacing, alignment, ruling, and shading. More detailed changes can be made from the Table menus such as cell height, page breaks, and straddled or rotated cells. Text contained in tables is styled by paragraph and character styles.

Once you’ve precisely defined the format specification for a table using the Table Designer, you can associate such a specification with a table tag that becomes part of the format catalog, available for reuse in the same document or for import into other documents.

Variables and conditional formatting

Variables let you reuse a single piece of information throughout a document. When you—or an automated process—update the variable value, the change is automatically reflected across the entire document. There are two types of variables in FrameMaker: user variables (user defined) and system variables (FrameMaker defaults). System variables can include the current page number, the current date, or the total number of pages in the current document.

Running headers and footers can use such general system variables, or can use twelve system variables provided specifically for running headers and footers. Such variables can refer to paragraphs with particular tags or marker text. In structured documents, running header and footer variables can refer to the values of elements or attributes.

FrameMaker provides different ways to format document objects depending on conditions. In a structured FrameMaker document, for example, you can set relationships where formatting changes based on attribute values and the context of a given element in the overall document structure.

With its conditional text feature, FrameMaker goes beyond formatting to let you actually include or exclude portions of the document based on conditions. In a database publishing application, this can provide customization and personalization of documents.

Markers

Markers are used by FrameMaker to create named locations in documents in order to provide capabilities such as document navigation, indexing, and autogeneration of lists. Markers establish named locations in documents: When documents reflow or repaginate, markers enable references to these locations, such as index entries or cross-references, to automatically reference the correct new location in the document.

Cross-references aid in the navigation of printed materials. Rather than explicitly write “see page 43” in a document that will be edited and may repaginate, you can use cross-reference markers to allow such references to be automatically updated to reflect the correct page. Cross-references can be of two varieties:

- Paragraph cross-references, which reference an entire paragraph
- Spot cross-references, which reference a specific location within a paragraph

Indexing a document involves creating index markers, which then identify locations in automatically generated indexes. Indexes can be of multiple levels; it is possible to use different types of markers to create multiple indexes (for example, a subject index and an author index).

TOCs and other forms of generated lists can also use markers. For example, you can add to an automatically created TOC specific additional locations referred to by markers.

While hyperlinks can be automatically created in indexes and generated lists, you can also manually create specific hyperlink references to named locations specified with markers.

Book handling

The book feature in FrameMaker lets you group multiple FrameMaker documents into a single book, which can be printed as a single document. Files such as TOCs and indexes can be automatically generated to treat the documents contained in a book as one entity, while page numbering can increment in relation to the book rather than an individual document. Automatically generating TOCs and indexes can be useful in aggregating content from multiple sources in the course of database publishing. The cross-referencing feature in FrameMaker allows you to identify and target references in the same or across multiple book chapters. Because this function is automated, cross-referenced locations are current each time the book is generated and updated.

Style management

FrameMaker has robust style management capabilities. Styles persist in format catalogs so they can be reused, as can objects from reference pages that use those styles. Techniques of applying style are tailored to meet the needs of specific document objects.

To stylize cross-references, for example, FrameMaker provides default formats (for example, “Table & Page” or “See Heading & Page”). You can also format your own cross-reference styles with “building block” syntax definitions.

Once you have autogenerated lists in FrameMaker (for example, Lists of Figures and Tables, TOCs, and indexes), you can stylize the generated paragraph style tags through the Paragraph Designer in the same way as with other style tags. Reference Page syntax showing the structure of the autogenerated lists can also be arranged and formatted.

Autonumbering features

FrameMaker offers powerful autonumbering features of individual document objects such as paragraphs (for example, numbered lists and lists starting with bullets or other special characters), headings, page numbers (including static section prefixes or suffixes), and figures. In a book structure, sections and chapters may be autonumbered using the Numbering Properties menu. This latter feature is most valuable when sections or chapters have been rearranged—FrameMaker will automatically reassign section and chapter numbers to match the order of the chapters in the book.

Benefits of FrameMaker

FrameMaker offers some unique advantages. Although it is well known for its capability to handle long, complex documents with high throughput and reliability, it also benefits from tight integration of the desktop and server products as well as its cross-platform capability. FrameMaker lets you format XML for print and Adobe PDF output without extensive coding. FrameMaker provides accessibility features for the benefit of both authors and end users.

Document handling of any size or complexity

FrameMaker can handle industrial-strength formatting and provide fast throughput, complex formatting, and the capability to produce very large documents.

Capability	Example
Mission-critical stability	Half of the world's top 20 banks use FrameMaker for mission-critical publishing, typically producing approximately 10,000 documents per hour.
Long document support	A single FrameMaker document can include millions of paragraphs/images/tables, thousands of different master pages, and tens of thousands of variables and cross-references.
Powerful table handling	A single table can contain millions of cells. Table cells can contain multiple images and multiple tables, nested to any level.
Robust indexing capability	20,000-page catalogs can contain multiple, complex indexes and hundreds of thousands of cross-references.
Dynamic book features	A FrameMaker book may contain hundreds of documents, each of any length, including hundreds of tables of contents and indexes.

Mission-critical stability

In addition to its reputation for volume, throughput, and complexity of its formatting engine, FrameMaker is extremely reliable and stable. For these reasons, institutions such as banks depend on FrameMaker for the accuracy and timely delivery of their information.

Template design without extensive XML coding

With FrameMaker, defining the way your XML will be presented does not require extensive coding. It can be done entirely from a user-friendly graphical interface. Visual tools such as the Paragraph Designer and Character Designer are more intuitive than hand coding; the ability to immediately view the results of your formatting with its WYSIWYG XML editing capability means faster and more accurate results.

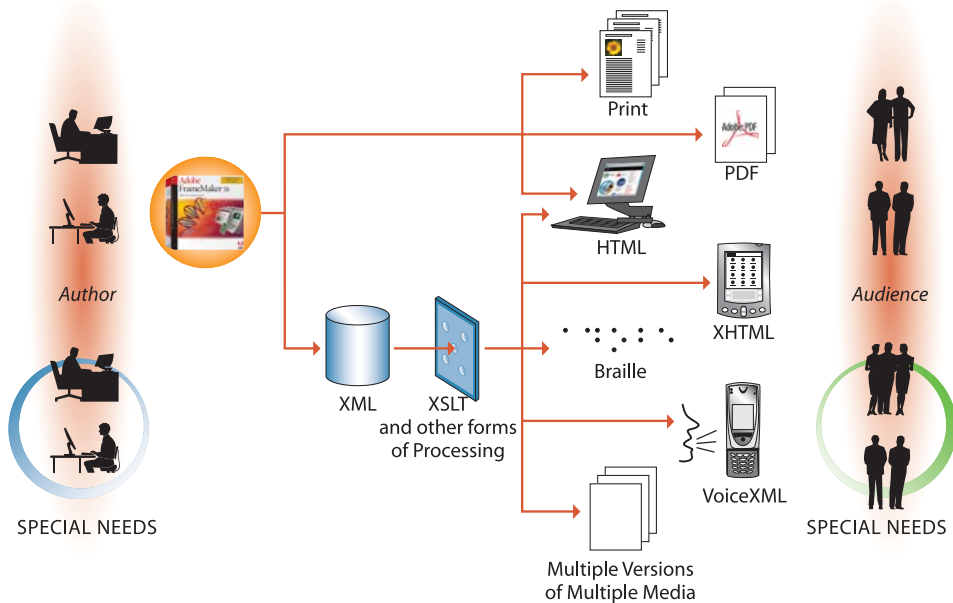
Accessibility features

FrameMaker provides accessibility features for both authors and consumers of content. By making your authoring process accessible, you can involve the full talent base of your organization; by making your content accessible, you can reach the widest possible audience or target market.

For authors, FrameMaker provides tight integration with the accessibility features of Microsoft Windows® 2000 and Windows XP. With these features, users can leverage a zoomed or high-contrast view of the document on which they are working, use keyboard shortcuts or have the FrameMaker screen read to them with a screen reader.

For those accessing content, the XML output capabilities of FrameMaker and tight integration with PDF enable the content delivery process to fully include those with special needs. XML provides the capability to deliver the same content to multiple formats and media such as Braille, VoiceXML, and Extensible Hypertext Markup Language (XHTML) browsers. Adobe PDF offers a wide range of accessibility features including tagged PDF, which lets you reflow your documents' content into devices with variable screen font sizes, or use a screen reader for Windows.

Additionally, the Object Attributes dialog lets you assign alternative text for graphic objects. The alternative text that you provide can pass through to resulting Adobe PDF, XML, or other forms of output.



Both authors and consumers of information may have special needs.

Cross-platform, desktop and server

FrameMaker is available on Windows, Macintosh, and UNIX® platforms as a desktop application. Its cross-platform support lets authors, template designers, and application developers work together across very different environments.

With the introduction of FrameMaker Server, FrameMaker can also be used on Windows and UNIX as a server application. The extreme flexibility of FrameMaker allows for publishing workflows that leverage the full range of skill sets across your organization.

It is important to note that while the desktop version of FrameMaker has powerful capabilities, it is not licensed for use in a server context. For clarification of the licensing requirements for the desktop version of FrameMaker, and a definition of those uses for which FrameMaker Server is required, consult the FrameMaker home page at www.adobe.com/products/frameMaker/.

For detailed information about FrameMaker Server licensing, consult the FrameMaker Server home page at www.adobe.com/products/fmserver/.

Implementing a FrameMaker solution

FrameMaker Server provides a publishing engine that requires integration into a solution before it can be used. The FrameMaker Server publishing engine can be integrated into publishing systems in a number of ways, including:

- Building a solution with FrameMaker Server
- Using Miramo
- Using FML PatternStream

Building a solution with FrameMaker Server

Building a solution without third-party tools typically requires customization by means of the FDK. There are three principal methods of leveraging the power of FrameMaker Server in such custom-built solutions:

- Using the XML capabilities of FrameMaker
- Using MIF
- Using the FDK to directly control the FrameMaker formatting engine

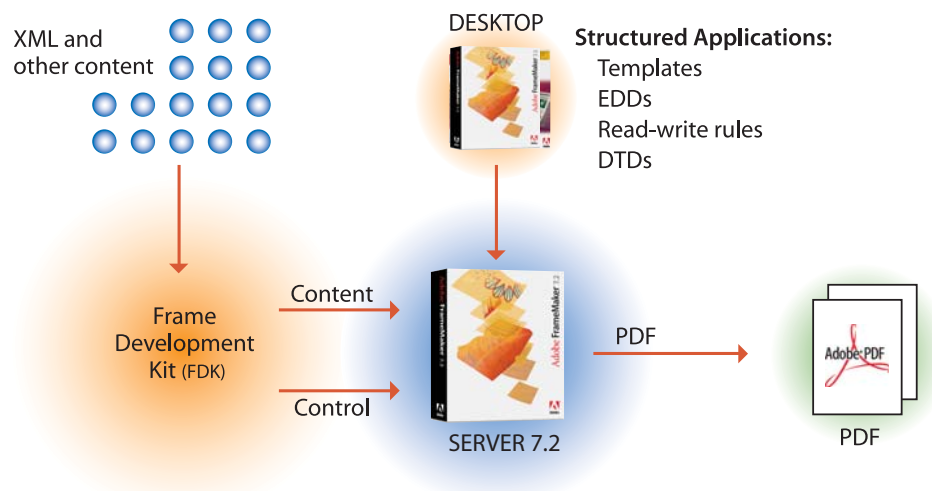
The best method for a particular publishing need depends on factors such as the nature of the data sources, the type of desired output, and the role of the publishing application in overall workflow and system architecture.

Using the XML capabilities of FrameMaker

The FrameMaker desktop product provides a robust structured authoring environment for XML and a means of formatting XML for high-quality print and PDF output. Its structured authoring capabilities allow you to define context-determined styles to XML via templates developed in a WYSIWYG environment. All of the XML formatting capability of this desktop product can be used with FrameMaker Server in server-side publishing applications.

By means of *structured applications* defined in the desktop product, you can specify an association between XML DTDs, FrameMaker templates, and other files required to format XML. Central to the formatting process are templates, which are FrameMaker documents that store document properties such as layout, formatting, and structure. In a structured application, templates include EDDs, which define elements, document structure, and context-dependent formatting of structured XML content. Structured applications can also reference *read-write rules*, which specify how XML objects are mapped to FrameMaker document objects such as graphics, tables and footnotes, as well as import and export specifications for graphics.

With this method of server-side publishing, XML that conforms to the same XML DTD of the FrameMaker structured application is streamed into FrameMaker under control of a lightweight FDK client. The source XML is formatted for print or PDF output according to the rules of the structured application. As long as the source XML continues to conform to the same DTD, formatting changes can be made independent of changes to the XML. Such a template-driven workflow can provide a powerful means of presenting XML content.



Templates developed with the desktop version of FrameMaker can be used to format XML with FrameMaker Server.

While XML files and XML repositories make a natural source of content, data housed in non-XML formats such as relational tables can also be converted to XML by means of intermediate programs or enhanced functionality of the database itself. There is an increasing degree of XML support in databases and database query tools.

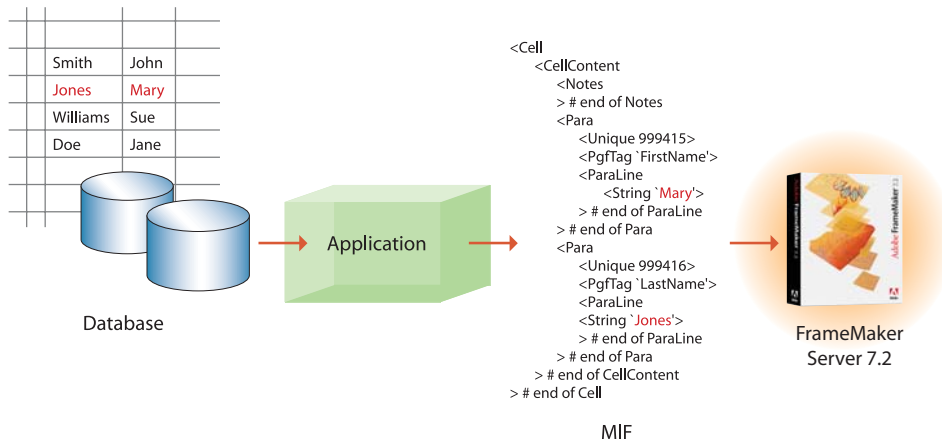
Using MIF

The capability of FrameMaker to save and load files in MIF (a text-based, easily parsed format) provides another way of using FrameMaker Server in database publishing applications. The MIF language and syntax are documented in the *Adobe MIF Reference*, which is provided in PDF form with FrameMaker. MIF provides an excellent intermediate format for manipulating FrameMaker documents programmatically in server-based applications.

In application architectures centered around MIF, a database publishing application typically extracts database content and wraps it with appropriate MIF code for automatic publishing. It is also possible to translate application-generated data to MIF, or to transform source XML files to MIF via XSLT.

It is not a requirement to dynamically generate every component of MIF input. MIF provides an include facility, so static components such as format catalogs can reside on the server and be referenced in the MIF that streams in rather than redundantly passed for each document.

As with FrameMaker Server applications built around the XML capabilities of FrameMaker, applications that stream MIF into FrameMaker Server can be automated by means of a lightweight FDK client. In this case, however, formatting for published output is already explicitly defined by the MIF itself, requiring less processing on the server.

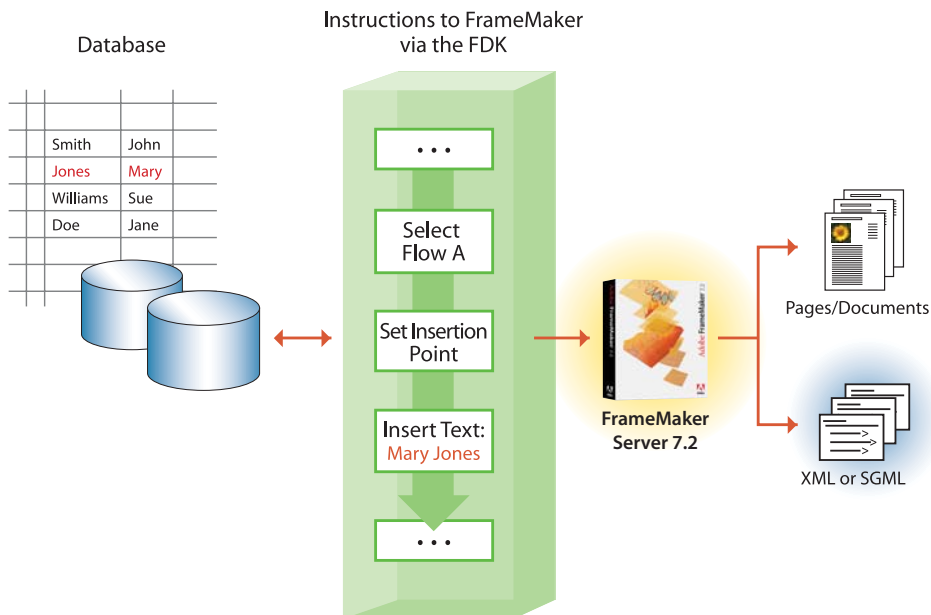


Streaming MIF into FrameMaker typically involves wrapping source data in MIF formatting instructions.

Using the FDK

Another method of database publishing built around FrameMaker Server involves using the FDK client in a more central capacity than it is used with XML- and MIF-based applications. The FDK client can completely control the FrameMaker Server application, including management of interface with external data sources and content destinations, creation and manipulation of FrameMaker document objects, and publishing either paginated pages and documents or export data such as XML or Standard Generalized Markup Language (SGML).

In such an architecture, the FDK establishes a connection to and queries data sources (which can include databases, XML sources or application-generated data), then programmatically composes and publishes documents based on the source content.



An FDK client can connect to data sources and control the FrameMaker Server publishing engine.

Using Miramo

Miramo, a third-party solution available from Datazone, uses FrameMaker both as a GUI template design tool and as a hidden pagination engine to produce documents at a rate of tens of thousands of pages per hour, or over a million pages overnight. Document types range from one- to twenty-page customer statements, produced in high volume and at high speed, through to enormous, multipage documents such as encyclopedias, directories, and industrial catalogs.

Miramo-created documents can be highly dynamic in layout and content. Under programmatic control, data-driven content formatting can produce multiple results from a single homogeneous input stream, with variations ranging from very granular (for example, minor formatting nuances in response to subtle variations in the source data) to global (for example, applying an entirely different document template based on flags in the source data).

A single Miramo host application can be accessed in multiple ways, ranging from use as an on-demand process triggered from user action in a web browser to unattended batch generation of documents and completely automated integration with other systems.

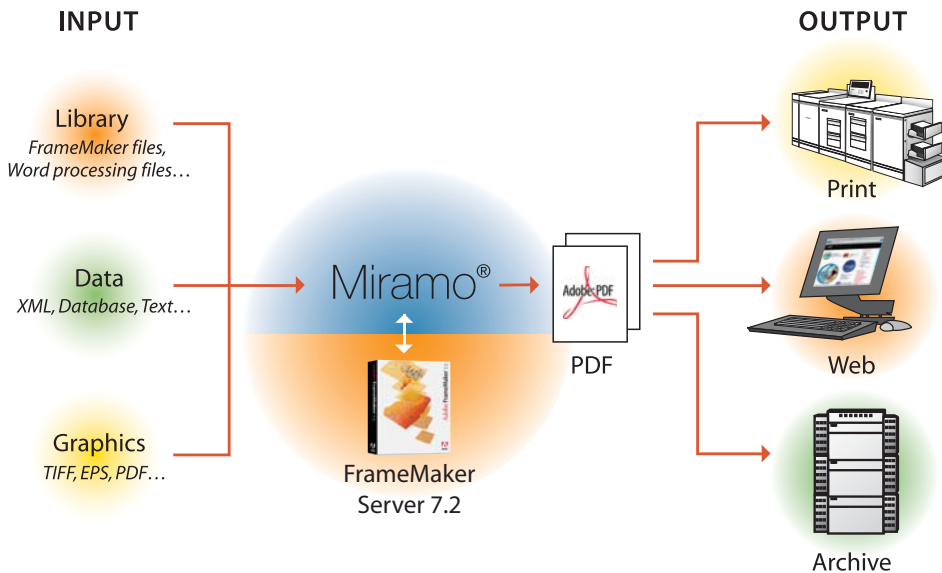
Designing FrameMaker templates for use with Miramo

The FrameMaker GUI provides the easiest way to design complex page layout templates for use with Miramo. Dynamic, data-driven formatting variations to the template can be programmatically controlled by Miramo. This includes all aspects of a FrameMaker template (for example, paragraph, font, table and variable catalogs). Miramo can dynamically define and apply master page layouts. An existing template's definitions can be used as is, or Miramo can apply data-driven formatting overrides on the fly. It can also define and add new formats to the catalogs of newly created documents.

Data and content sources

While Miramo is most often used with input generated by a database query, Miramo input can come from a wide variety of sources, including static text files, .csv files, legacy system or typesetting files, XML, HTML, SGML, or component FrameMaker files. Any of these data sources can be intermingled with database-generated data or any of the valid FrameMaker input formats (for example, word processing files).

Miramo can dynamically create and send formatted data to an external application for processing, then incorporate the result into the Miramo document. Possible examples would be formatting complex equations, scientific charts or graphs, or third-party graphics applications.



Miramo and FrameMaker Server provide a powerful, server-based publishing application.

Working with XML

With `mmxslt` (the built-in Miramo XSLT processor), Miramo can transform XML-encoded data streams into fully formatted output (for example, .fm or PDF files, or any of the FrameMaker formats). In addition, Miramo can divert input data through any external validation or parsing process desired prior to resuming processing. The input may include a reference to the Miramo DTD or to any external DTD.

Miramo's XML-handling capability, in combination with the powerful formatting engine of FrameMaker, lets you add sophisticated layout and document formatting to any XML-based application.

Working with `mmpp`

`mmpp` (the Miramo macro preprocessor) can be used as an input filter to transform incoming data streams for subsequent Miramo processing. `mmpp` supports macro arguments, file inclusion, reading and processing external files line by line, system commands, expression evaluation, conditional output, 'for' loops, variable definitions, 'regular expression' matching and substitution, as well as a number of powerful string and arithmetic functions.

Automating the publishing process

Just as the term "publishing" has been redefined to include content delivery in multiple media (for example, paper, eBooks, wireless data, and HTML), the concept of "publishing automation" has also been expanded to include a wide range of automation requirements.

With a publishing framework built around FrameMaker and Miramo, implementations can range from multifocus, web-based, on-demand systems for catalogs or personalized account summary generation to highly sophisticated, single-focus background processing for banking and financial systems. With Miramo, flexibility in content creation and delivery media does not come at the price of a concomitant complexity or rigidity in implementation. Interfaces to a Miramo-based process can be developed with virtually any tool (for example, VB, Java, WSH, ASP, and PHP) to accommodate any degree of ad hoc input or control required.

Using FML `PatternStream`

`PatternStream`, a third-party solution available from FML, provides a complete interactive environment for building data-driven publications. Using a GUI, the user can manipulate objects that define how data will be accessed and formatted. `PatternStream` also provides a database access layer that acts as a direct connection between data sources and page layout and composition engine in FrameMaker.

`PatternStream` architecture

`PatternStream` is implemented as an extension of FrameMaker, and uses FrameMaker templates to provide a catalog of style resources. The `PatternStream` system is designed to enable rapid setup of data-driven publishing projects by means of a visual programming language that allows users to manipulate relocatable objects. No external scripts or programs are needed to extract the data and no intermediate markup files are required for processing.

`PatternStream` feeds data from a series of nested queries directly into the FrameMaker formatting engine, enabling users to quickly see the results of changes they make to the document's design and layout logic in the course of iterative development.

The PSet Hierarchy

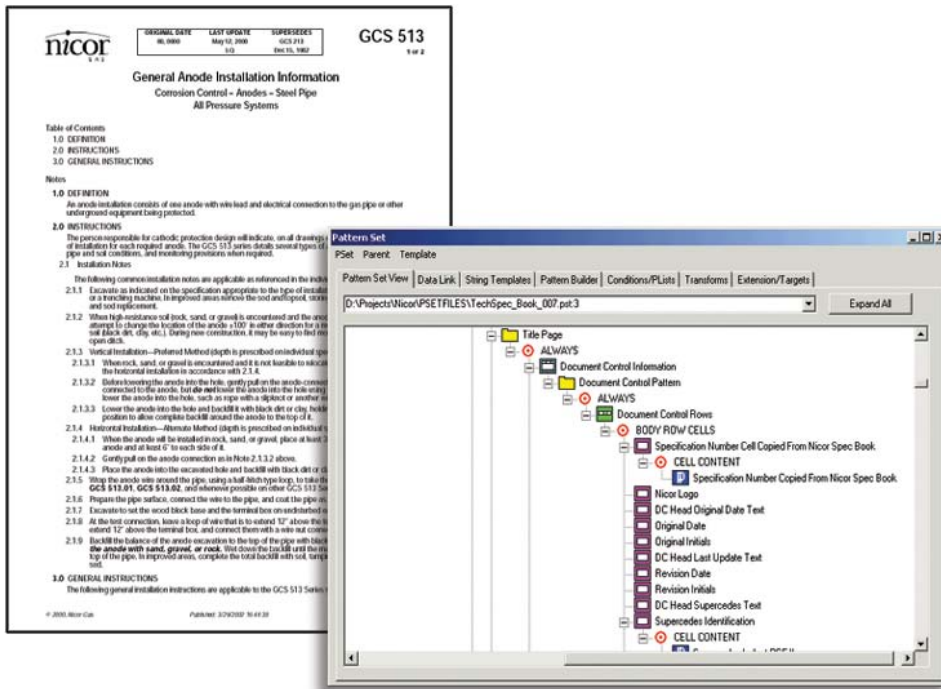
In documents used to present structured data, the layout logic often reflects the underlying data structure. For data-driven documents, this layout logic will be reflected as a repeating sequence of nested patterns of structural elements and text. The structure of these nested patterns can be represented as a data hierarchy. Documents can often be viewed as a logical hierarchy of structural elements. An example of a document being viewed in this manner would be the DTD of an XML document type.

However, the levels in the data hierarchy may not match exactly with the levels in the document hierarchy. `PatternStream` is built around a paradigm that uses both of these natural ways of viewing the output, and provides a means of merging these two different views together. The method for doing this is a construct called the PSet Hierarchy.

This PSet Hierarchy is composed of an interleaved, nested structure of pattern objects and target objects. Each pattern object has an associated query; together, these patterns define the data hierarchy. The target objects contained in each pattern are used to generate structural elements such as paragraphs and tables. The number of times a pattern uses the attached target objects to generate structural content is determined by the immediate results of its associated query.

Connecting to data sources

FML PatternStream is a data-centric application. It allows direct connection to any Open Database Connectivity (ODBC)-compliant database, as well as flat files and custom data sources. You can connect to multiple data sources simultaneously. The queries from these different data sources can be nested together with no preprocessing or intermediate markup files, providing flexibility of data access and layout strategy.



PatternStream lets users manipulate pattern and target objects through an intuitive visual interface.

Object classes in the PatternStream application

FML PatternStream offers a rich collection of objects that can be used to build complex, data-driven documents:

- **Patterns:** used to represent a collection of structural elements (for example, paragraphs and table rows) that repeat. Patterns are associated with queries and contain targets.
- **Targets:** used to create the structural elements (for example, tables and paragraphs) associated with patterns. Targets do not directly create textural content; however, some targets can use attached string templates for this purpose.
- **Queries:** used to extract data from an outside data source. Patterns must be associated with a query object to be invoked. A query can be executed as many times as the logic of the document layout requires.
- **Variables:** used to hold the data extracted by queries. They can also be assigned values programmatically.
- **Connection objects:** used to provide connections for data sources.
- **String Templates:** used to create an instance of text. String Templates contain multiple segments, which can represent variable data, constant strings or a wide range of document objects.
- **Transforms:** used to change the values of variables (for example, computation or string manipulation).

- *Conditions*: used to make the generation of structural elements in the output document conditional on whether various data criteria are met.
- *Parameter Lists*: used for specific purposes (for example, passing values to a FrameScript).
- *Extensions*: miscellaneous objects or extensions to the basic PatternStream application, including customized plug-ins written in C or in FrameScript, associative arrays, and reusable subtrees.

Next steps

If, after reading this paper, you have concluded that your organization could improve its database publishing techniques, you may benefit from a critical assessment of your needs and an evaluation of database publishing tools.

Database publishing needs assessment

Depending on industry, company size and specific demands for published output, different companies face very different publishing needs. When assessing these varied needs, it is useful to consider both the inputs to the system and the desired qualities of published output.

Issues to consider regarding inputs to a database publishing system include:

- The range of data sources (for example, multiple databases, XML data sources and flat files)
- The formats of integrated graphics (for example, SVG, Tagged Image File Format [TIFF] and EPS)
- The formats of those document sources that are not data-generated (for example, static front matter)
- The likelihood of data source consistency over time

Issues to consider regarding published output include:

- The range of desired output media and formats (for example, Adobe PDF, HTML, and XML)
- The required throughput of the system (for example, documents per hour for each output format)
- The scope of required personalization of the system
- The complexity of document formatting
- The degree of advanced composition requirements such as autogenerated indexes and TOCs

Database publishing solutions based on FrameMaker can accommodate a diverse array of needs. FrameMaker tends to be most valuable when there is a wide range of input and/or publishing formats, demand for high throughput (for example, quickly generated documents respond to web queries or batch composition of many documents in a short time frame) or complex composition and/or personalization demands. FrameMaker also offers particularly powerful capability for organizations that maintain their documents as XML.

FrameMaker evaluation

For more details about FrameMaker, including information on the availability of evaluation versions of the program, please go to the FrameMaker product Web page at www.adobe.com/products/framemaker.

Checklist for a database publishing solution

When evaluating whether it's time to upgrade your publishing solution, consider prospective applications using the questions in the following table:

Question	FrameMaker	Your Application
Can you easily publish from a single source to high-quality print, PDF, HTML and XML formats?	X	
Can you easily publish long documents with complex formatting and automatically generated indexes and tables of contents?	X	
Can you effectively style XML for delivery to print and PDF?	X	
Are hyperlinks and bookmarks automatically generated when publishing to PDF as well as to web formats?	X	
Can you produce high document throughput with confidence in your application's mission-critical reliability and stability?	X	
Can you leverage a proven, fully featured document object model for formatting your documents?	X	
Can your publishing system save out and read in a complete representation of documents in an easily parsed, text-based format with no loss of information?	X	
Can you precisely control the location of graphic objects in relation to text, with proper relocation of such objects when documents are reflowed or repaginated?	X	
Can you leverage a template-driven workflow in which content is maintained independently from formatting?	X	
Can you save and reuse formatting styles for a complete range of document objects, including paragraph, character and table styles?	X	
Can you reliably format tables with complex structure and detailed formatting spanning many pages?	X	
Is your document production tightly integrated as a fully automated process within your business' other mission-critical applications?	X	

Conclusion

Database publishing is a critical need across many industries with a wide range of different kinds of database publishing needs. With advances in database technology, an increasing number of media and formats for content, and increased requirements for real-time access to information, demands on database publishing systems have increased. With enhanced XML support, new accessibility features, tight integration with the latest version of Adobe PDF, and support for new graphic formats such as SVG, FrameMaker continues to evolve in meeting these demands.

Whether your enterprise publishes single-page invoices or large, complex documents spanning thousands of pages; whether your data sources are primarily XML, database or other formats; and whether you deliver more to web or print output, FrameMaker warrants a thorough evaluation as the central component of a database publishing solution.

Appendix: Resources

www.adobe.com/products/fmserver/	FrameMaker Server product page
www.adobe.com/products/framemaker/	FrameMaker product page
http://access.adobe.com/	accessibility information about Adobe products
www.miramo.com/	Datazone Miramo home page (worldwide)
www.axialinfo.com/	a US Miramo source
www.patternstream.com/	FML PatternStream home page



Adobe Systems Incorporated • 345 Park Avenue, San Jose, CA 95110-2704 USA • www.adobe.com

Adobe, the Adobe logo, the Adobe PDF logo, Clearly Adobe Imaging, the Clearly Adobe Imaging logo, Frame, FrameMaker and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Mac and Macintosh are trademarks of Apple Computer, Inc., registered in the United States and other countries. Sun Microsystems, Inc. is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. UNIX is a trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners. SVG is a trademark of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions MIT, INRIA and Keio.

© 2005 Adobe Systems Incorporated. All rights reserved. Printed in the USA. 8/05

