

---



**datafloat**

Powering Global E-Commerce

# Technical Overview

November 2000

IDC Global, Inc  
[www.idcglobal.com](http://www.idcglobal.com)

---

## Contents

<b>INTRODUCTION</b>	<b>1</b>
<b>DATAFLOAT PRINCIPLES</b>	<b>2</b>
<b>1. Globalization</b>	<b>2</b>
Multilingual Databases	2
Multilingual Data Processing	3
Multilingual Static Content	3
Example: Converting an Existing Web Site to Multilingual	3
Example: Client Side Language Switching	3
Language Translation	3
Example: Normalizing Data to Save Future Translation Requirements	4
Localization of Format and Measurements	4
<b>2. Client Side Data Processing</b>	<b>5</b>
Example: Search and Display Results Using dataFloat	5
Example: Retail Catalogue	6
<b>3. Multi-Device</b>	<b>6</b>
<b>4. Flexibility</b>	<b>6</b>
<b>5. Scalability</b>	<b>6</b>
<b>DATAFLOAT DATABASE FORMAT: "DATAFLOAT METADATA"</b>	<b>8</b>
<b>1. dataFloat Metadata is Compact</b>	<b>8</b>
Example: dataFloat Compact Data Structure	8
<b>2. dataFloat Metadata is Multilingual</b>	<b>9</b>
Example: How to Add a New Language using dataFloat Administrator	9
<b>DATAFLOAT SOFTWARE COMPONENTS</b>	<b>10</b>
<b>1. dataFloat database Initialization scripts</b>	<b>10</b>
<b>2. dataFloat Code Libraries</b>	<b>10</b>
Target Client	10
Data Processing	10
Data Format	10
Application Code	10
XML Model	11
JavaScript Model	12
Server Side Parsing Model (HTML and WAP)	12
Palm O/S Model	12
Windows Pocket PC	12
Off-Line Applications	13
Example: Off-Line Database for B2B Applications	13

Example: Email Catalogues for Off-Line Browsing	13
Multilingual Functionality	13
<b>3. dataFloat Administration Tools</b>	<b>14</b>
dataFloat Administrator	14
MS Front Page Style Sheet Add-In	14
dataFloat Translation Manager	15
<b>BUILDING APPLICATIONS WITH DATAFLOAT</b>	<b>16</b>
<b>FURTHER INFORMATION</b>	<b>17</b>
USA	17
Europe	17
Canada	17
Indonesia	17

## Introduction

dataFloat is an XML data processing platform for building database generated Internet and Intranet applications. dataFloat provides the data structure and the software components necessary to build Web, WAP and Palm e-commerce applications.

dataFloat addresses a number of issues currently faced by developers when building database generated Internet applications. These are as follows:

1. Globalization of Internet applications
2. Client side and off-line data processing
3. Multi-device data access and rendering
4. Flexibility
5. Scalability

dataFloat provides an integrated platform to build applications based on these principles.

The dataFloat package consists of three components:

1. A "row-wise" data structure which is compact and multilingual;
2. dataFloat software components; and
3. dataFloat tools, including
  - dataFloat Administrator: to build and manage dataFloat databases
  - dataFloat Translation Manager: to efficient translate dataFloat applications
  - dataFloat XSL Authoring Tool: to build style sheets for dataFloat databases.

The combination of these three elements provides developers with a platform for rapidly building multilingual, multi-device applications, which incorporate client side processing, flexibility and are highly scalable.

This document provides an overview of the dataFloat platform. It begins by outlining the principles on which dataFloat is based. It then describes each of the components above, which make up the dataFloat package. Finally it describes the methodology of building an Internet and Intranet applications using dataFloat.

## dataFloat Principles

### 1. Globalization

The dataFloat platform allows you to build globalized Internet applications. Globalization consists of two key elements, which have been defined by the Localization Industry Standards Association (LISA) as follows.

**Internationalization** is the process of generalizing a product so that it can handle multiple languages and cultural conventions without the need for redesign.

**Localization** is the process of adapting a product to meet the language, cultural and other requirements of a specific target environment or market (a "locale").

dataFloat allows developers to build applications, which are internationalized and able to be localized in a highly efficient and effective way by providing the following functionality:

1. Development and management of multilingual databases
2. Multilingual data processing code
3. Management of multilingual static content
4. Format localization
5. Currency and Measurement Localization

dataFloat supports both double byte Unicode and right to left languages.

### Multilingual Databases

The dataFloat data structure allows you to store multilingual data in one database, which means you can develop and maintain internationalized databases efficiently and effectively.

There are two common ways of creating multilingual data stores. The first is to replicate the database in each language. This method inherently has a number of problems. Firstly it is an inefficient method of storage because there is a large amount of redundant data. For Internet based applications this means slow download times and/or increased bandwidth requirements. Secondly as the number of languages increase, synchronization of the databases becomes difficult and time consuming. These types of systems also become very inflexible. For example, any change in data structure will need to be repeated for each database, and for all code accessing each database.

The second method involves storing the multilingual data in one database and building language dictionaries. This involves storing the data as non-language specific code and storing the language specific labels separately. This method solves many of the problems associated with replicated databases, however can often involve a large amount of detailed data design, especially if building complex data structures. This solution will also require more effort for application development as data rendering requires matching of the raw data to language specific labels.

dataFloat uses a form of language dictionaries set out in a row-wise structure. It allows for rapid development and more efficient management of applications. Using the dataFloat Administrator tool developers can set up multilingual data structures efficiently and without the need for detailed data design work you can set up very complex data structures including multiple record types, multiple data trees and embedded data objects. dataFloat automatically sets up all the language dictionary structures required for the database and defines the complex relationships behind these objects automatically.

dataFloat provides very fast search speed. This has been achieved by a combination of indexing and tuning the dataFloat search algorithms to maximize speed. We believe that the search speed of a dataFloat database is as fast as a well built relational database.

## Multilingual Data Processing

dataFloat manages all the data processing and rendering requirements of a multilingual database. This includes all language dictionary matching, multilingual queries, formatting, font requirements, and multi-device issues.

dataFloat allows developers to build multilingual functionality into the front end of applications. dataFloat code carries out all the multilingual aspects of displaying and querying multilingual data. New languages and alterations to the data structure are automatically recognized by the dataFloat code.

## Multilingual Static Content

Static multilingual site content is content, which is not database driven. dataFloat separates the html into a style sheet and a text file. The text is then translated into multiple languages, but the style sheet remains the same for each language.

dataFloat separates pages into XML and XSL, or into text files and ASP/PHP. Which is best depends on the particular application and server platform.

### Example: Converting an Existing Web Site to Multilingual

dataFloat provides a multilingual “splitter”. This is an application, which automatically splits an html based web page into a style sheet and text file. This tool allows for rapid conversion of existing web sites into multilingual sites. dataFloat functions can then be used to quickly add language controls into the site.

dataFloat provides the server and client side code to allow efficient switching between languages. A language “control panel” can be incorporated into an application, and/or language preferences can be stored with user login information.

dataFloat allows client side language switching by allowing multilingual text and multilingual data to be downloaded to the client. Language switching can then occur without the need for a return trip to the server, or for the data to be re-downloaded.

### Example: Client Side Language Switching

dataFloat can download data and/or static content in multiple languages. This means that you can quickly switch between languages using client side processing.

It is particularly useful for front pages, at a point before the system knows which language the user will want to view. You can for example use mouse over hot spots to rapidly change the language.

## Language Translation

The dataFloat Translation Manager allows for human and machine translation to be performed efficiently. Translators can log in remotely to a dataFloat powered web site and will be automatically directed to data requiring translation. This will be automatically integrated into the web site.

The dataFloat Translation manager allows you to translate the data, metadata and static content, all from a single application. Translators can see content in its context and do not need any programming or database knowledge in order to translate the data.

The translation manager can also be integrated with a machine translation tool. Translators can then check and correct the machine translated content.

dataFloat allows you to normalize your multilingual data for efficient on-going management. Normalized content is that which is generated from a pick list of data objects. Normalized content can be stripped of all the language content, and the data objects translated only once. This means that new and updated records do not need to be translated. The dataFloat administrator allows you to build complex and diverse data structures, so that you can move as much of the data away from free text fields to normalized fields. This can significantly reduce the on-going translation requirements of an application.

#### Example: Normalizing Data to Save Future Translation Requirements

In the real estate industry, much of the descriptive data was traditionally put in a free text description field. This is because of the diverse range of features involved with real estate. Using dataFloat we were able to move many of these diverse features into normalized multilingual data fields. This has significantly reduced the amount of translation necessary.

Without dataFloat, the extra overhead of a large data structure often makes this kind of operation impracticable, especially when incorporating multilingual dictionary structures.

#### Localization of Format and Measurements

dataFloat uses the XML model to separate data from formatting code. The data is downloaded in a compact XML format, while the rendering code is downloaded in an XSL style sheet. You can define multiple style sheets for one set of data. This means that data can be formatted differently for each locale to ensure that it has the correct “cultural feel”. This is done without having to rewrite the data processing code for each locale.

The dataFloat data structure and data processing code also allows for definition of multiple currencies and measurement models.

## 2. Client Side Data Processing

dataFloat allows you to build Internet and intranet applications which incorporate client side data processing and rendering. Client side processing has a number of significant benefits including faster, more interactive sites and large increases in scalability. The traditional method of delivering database-generated content to web pages involves static html. All data processing and rendering occurs on the server side, with a static html page being delivered to the client. The client machine can do nothing but display the html. If there is any user interaction with the data, a request is sent back to the server for a new page. Even a small adjustment, for example a resort, involves the server reprocessing the page and resending the html including all the data. This can often happen several times as a user browses a set of results.

The dataFloat client side-processing model is far more efficient. The server transfers to the users machine (the "client") a piece of compact data, metadata and processing code. The processing code instructs the client how to render the data according to the metadata. The data can then be reused without accessing the server again. The metadata and processing code can also be reused and thus are only sent once per session.

Client side data processing takes advantage of the fact that each attached client machine connected to a server brings a new processor and more memory to the system. By using the client machine processing power rather than the server, it reduces pressure on servers and bandwidth allowing for highly scalable Internet applications (see Section: Scalability). It means significantly faster and more user interactive web sites.

### Example: Search and Display Results Using dataFloat

When a search page is requested, the server delivers the metadata and style sheet, for rendering the search form, client side. This metadata is reused for displaying the results. After the user enters the search criteria, an XML search query is created client side. This search query is sent to the server for processing.

The server will process the query and return a set of compact data. This set will be a pre-defined number of records, say 100. The 100 records will be downloaded to the client along with the client side code. Once this is done, the 100 results can be browsed, reformatted, sorted, and re-searched without having to go back to the server.

You can build front end functionality to do things such as: display of the results 10 at a time, sort the data in different ways, display tables showing different fields, display individual records, create reports - all done client side without having to access the server again.

The dataFloat client side code will transform SQL queries into XML and send it to the server for processing. dataFloat returns the result set in compact XML format. The compact nature of dataFloat databases means that significantly more data can be downloaded to the client machine for processing without returning to the server. dataFloat code provides the functionality to access the compact data on the client and perform a large amount of client side data manipulation including browsing, reformatting, sorting, aggregating, refining searches and changing languages.

dataFloat can use pre-generated queries or sections of the database. A database may be split into downloadable sections for client side browsing. Use of pre-generated data pages will further decrease server processing and increase speed and scalability. For the most optimized solutions, the servers only job is to download these pre-generated files. All rendering is done on the client. However for many applications this is not possible and dataFloat allows you to develop mixed client side/server side applications, which are fast, interactive and highly scalable.

#### Example: Retail Catalogue

A retail catalogue can often be split up into easily identifiable sections. For example, a clothing catalogue may be split into men's and ladies' clothing, then split further into shirts, trousers, shoes etc...

A user can select the section required and download all of the catalogue data for that section. Any browsing of this section is then done on the client and can be highly interactive, fast, and requires no server processing.

Queries containing the data for each of these sections can be pre-generated, to further reduce server processing.

### *3. Multi-Device*

Internet applications often target a number of browser versions and devices. The extent to which client side processing can be carried out is dependant on the client browser or device. To optimize performance dataFloat generates code targeted to specific browsers and devices. Each device has its own code generator. Each code generator is hand tuned and tested. By using native code we can get the best performance out of each device. The data structures, user commands, and renderings can be specified and then code generated specific to each platform. dataFloat currently supports Web browsers, Palm Pilots and WAP devices. For more information on the models for each device see Section: dataFloat Code.

The dataFloat code allows for rapid deployment. Applications take a few man-months rather than man-years, and very little maintenance. Most maintenance can be handled through the dataFloat GUI Administrator, and other changes handled by style sheet editors, such as MS FrontPage and Macromedia Dreamweaver.

### *4. Flexibility*

Most database generated applications hard code the database metadata into the applications. This means that any changes in the data structure require changes to be made to the application code.

dataFloat does not hard code metadata, rather it sends reusable metadata to the application along with the data. dataFloat server or client side code then merges the metadata with the data received. This means that the data structure is flexible. dataFloat ensures that metadata changes do not break existing data or code, whether it is server or client side. Therefore you can alter the metadata without the need for large amounts of reprogramming. Alterations such as adding new fields, new record types and new languages can be done quickly and with a minimum of developer effort.

dataFloat also provides dynamic style sheets to allow you to build metadata dynamic Internet applications. These style sheets are generated from the metadata and can be inserted into a front end, formatted and functionality added. If developers do not want to use the dynamic style sheets, then dataFloat functions can be used to develop customized front ends from scratch.

### *5. Scalability*

Scalability can be defined as the ability to add consumers of resources while maintaining specified performance.

A counterexample in scalability is a "traditional" interactive web site. Static pages are served up in succession according to the users' responses. For example, a browse of 1000 records is

done 10 records at a time, each time a remote server is generating new code for the client. As the number of concurrent clients increases, the server takes almost all the extra load. The inherent bottlenecks in this system are server load and bandwidth. The trade off between data transfers, interactivity, and performance is severe.

dataFloat uses client side processing to increase performance and scalability. When data is requested from the server, a large data set is returned in a compacted format. Code that rarely changes, such as JavaScript functions and style sheets, is separated out from data. Such code is downloaded only when it changes on the server, or when the cache expires. The user can now browse, sort, search and reformat the data without any interaction with the server. Web sites are far more scalable when this method is used.

Even for data entry and modification, this system is more scalable. Much of the data modification resource requirements are used in data rendering and validations. Most of the validations can occur independent of any interactions with the server. Once data is submitted, it is transacted like any other system, but the servers are only involved in the transactions, and not in processing which is not required to be done by a centralized data store.

dataFloat can be used on load-balanced systems and it will considerably reduce the server requirements and bandwidth necessary for the system. For some applications, however dataFloat will do away with the need for “Web farms” and load balancers. Highly interactive applications with hundreds of users can run comfortably using one server.

For more information on Scalability see: *White Paper: dataFloat and Scalability*.

## dataFloat Database Format: "dataFloat Metadata"

dataFloat provides a very compact, flexible and multilingual data structure, which can be efficiently set up, maintained and optimized for each application.

### 1. dataFloat Metadata is Compact

The attribute of compactness has two components: the data itself and the model used to encode it. The model used to encode the data is also called "metadata". The richer the model is, the more compact the data. dataFloat uses a proprietary metadata, which is highly compactable. Our tests have shown dataFloat metadata to significantly out perform traditional relational databases.

#### Example: dataFloat Compact Data Structure

Comparing the size of data files stored using different data structures can see the compact nature of the dataFloat Metadata: (a) dataFloat compact XML, (b) a standard html table. The following shows the file size of 1,000 records from a live database application we have developed for one of our clients.

	dataFloat XML	HTML Table
File Size	2,638 KB	33,714 KB
% of HTML	7.8%	
File Size Multiple		12.8x

The dataFloat Metadata can reside on any modern database; currently we support MS SQL Server 7/2000 and Oracle 8i. Support for other database applications will be available soon.

dataFloat uses "application specific" metadata, rather than the "general purpose" models such as Lempel-Ziv and Huffman encoding. This means that for each application, an encoding scheme, which optimized to the particular application, is used.

The dataFloat data structure can be set up and maintained by the dataFloat Administrator. This is GUI tool, which allows developers to quickly and efficiently set up complex multilingual data structures, which are well structured and highly compactable. The dataFloat Administrator is also used for on-going maintenance of databases.

The dataFloat metadata is proprietary to dataFloat, however as part of a license to customers we allow access to the source structures and design documents. This means that our customers can customize dataFloat technology and have a large degree of control over their systems.

The level of compaction possible is also dependant on the data itself. The more normalized the data, the more compactly dataFloat can store it. Normalizing databases also has other benefits. The data becomes more accurately searchable and it reduces translation requirements. While it is up to the individual database designer to organize the data in a normalized fashion, dataFloat can help in this process. Due to the efficiency of setting up complex data structures, data designers can afford to move diverse information previously stored in free text fields into normalized fields.

## 2. dataFloat Metadata is Multilingual

The dataFloat data structure allows you to store multilingual data in one database, which means you can develop and maintain compact multilingual applications efficiently and effectively.

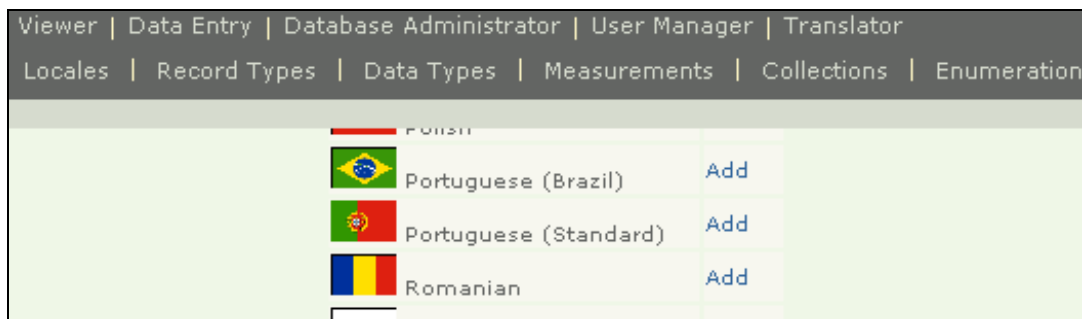
The dataFloat Administrator tool allows developers to set up multilingual data structures efficiently and without the need for detailed data design work. Using dataFloat you can set up very complex data structures including multiple record types, data trees and embedded data objects. dataFloat automatically sets up all the language dictionary structures required for the database. dataFloat supports both double byte Unicode and right to left languages.

Adding new languages is efficient. You can add a language with one click. dataFloat will automatically set up all the new data structures, languages dictionaries, data processing and rendering code necessary for the new language. This means that you can enter new markets very quickly and with a minimum amount of developer time and effort.

dataFloat systems are very flexible, the dynamic metadata model which dataFloat uses (see above) means that multilingual data structures can be altered without braking existing data or code. The dataFloat Administrator will automatically alter all the multilingual aspects of the data structure, and changes will flow through the application without substantial re-programming. This saves a considerable amount of developer effort, and ensures against data corruption.

### Example: How to Add a New Language using dataFloat Administrator

Use the dataFloat Administrator to choose "Locales". This will show a list of locales currently set up for the database, as well as a list of new locales to add. dataFloat currently supports the standard 76 locales. New locales can be added if necessary. Clicking on "Add" beside a new locale will set up all the data structures for that locale. dataFloat will also set up the files to handle the static content for the new locale.



Using the dataFloat Administrator you can also set up currency conversion and preferred measurement system for the new locale. The application is now ready for translation of data, metadata and static content. Using the dataFloat Translation Manager can then carry this out. Front end formatting for the new locale will need to be checked.

All front end, viewing and data entry functionality will automatically recognize the new locale, the application is ready for use in the new language.

## dataFloat Software Components

The dataFloat software components allow developers to build Internet applications using dataFloat databases.

dataFloat includes the following software components:

1. dataFloat database initialization scripts,
2. dataFloat code libraries,
3. dataFloat dynamic style sheets, and
4. dataFloat administration tools.

### *1. dataFloat database Initialization scripts*

The database initialization scripts set up the database structure (tables, stored procedures, relationships) necessary for a dataFloat application. These are:

1. the dataFloat meta tables, and
2. the dataFloat stored procedures.

The dataFloat Metadata tables are used to store the data in the dataFloat format. They are a set of standard tables, which define the data structure, data and language specific labels.

The dataFloat stored procedures carryout database functionality including storing data, querying data, creating compact XML result sets, creating XML metadata, creating indexes and carrying out other database maintenance and accessibility.

Currently dataFloat databases can be set up on MS SQL Server 7/2000 and Oracle 8i. Support for other databases is planned for the future dataFloat versions.

### *2. dataFloat Code Libraries*

The dataFloat code libraries are stored on the web server. dataFloat currently supports MS IIS 5. We are currently working on an Apache version, which we expect to release in Q1 2001.

The dataFloat code libraries carryout the following functionality:

1. Client side data processing,
2. Off-line capabilities, and
3. Multilingual functionality.

The dataFloat code library has been written using a number of technologies depending on the device for which it is targeted. dataFloat code is designed to execute using resources on the client device where ever possible. However where is this not possible due to the limitations of the client device, dataFloat code can be executed on the server for delivery of static content to the client.

dataFloat uses the following models, database formats and code languages depending on the target client device.

Target Client	Data Processing	Data Format	Application Code
XML browser	Client side	XML	XSL & JavaScript
JavaScript browser	Partial client side	HTML	JavaScript
HTML browser	Server side	HTML	-
Palm O/S	Client side	DB2	C++
WAP phone	Server side	WML	WMA
MS Pocket PC	Client side	XML	XSL/JavaScript

## XML Model

Where possible dataFloat uses XML for client side data processing. Using this model structured data and rendering code are separated into:

- a) An XML file which contains the dataFloat compact data,
- b) A dataFloat metadata file also in XML, and
- c) A style sheet, which contains the rendering code (XSL/JavaScript).

The compact nature of the dataFloat XML means significantly more data can be downloaded to the client machine for processing without returning to the server. dataFloat code provides the functionality to access the compact data on the client and perform a large amount of client side data manipulation including browsing, reformatting, sorting, aggregating, refining searches and changing languages.

Developers use dataFloat functions to build the functionality and front-end. Style sheets can be edited using web editors and dataFloat tags entered to mark where database generated content should be delivered for display the screen or other functionality. A FrontPage add-in is provided to insert dataFloat tags into style sheets.

Alternatively dataFloat also provides dynamic style sheets, which can be used to create query, result and report screens. The style sheets are formed by the metadata and are totally flexible. These style sheets can be edited and formatted using normal web site editors.

dataFloat provides the functionality to build complex queries. Queries are transformed into XML on the client and sent to the server for processing. The dataFloat stored procedures return compact XML result sets.

Non-database generated content and front end formatting is created using traditional Web technologies (e.g. HTML, Java, ASP, WAP, etc...). For multilingual sites, the static content is stored in XML and rendered using XSL in the chosen language.

The dataFloat XML model currently requires MS IE 4 and higher with the MSXML 3.0 parser update. Full support for Netscape 6.0 XML is currently being developed and will be released soon. The data can be rendered server side for older or non-XML supporting browsers.

In order to display data on a XML enabled web browser, the following steps are carried out:

1. Create database connection: For example use ASP to open an ODBC connection.
2. Call *getMetadata* dataFloat function: This produces an XML file containing the metadata for the database (often pre-generated). The XML file will be delivered to the browser and is used by the client side code to access the compact dataFloat XML.
3. Carryout query and produce compact XML result set (*XMLfromSQL* dataFloat function): dataFloat functions allow you to specify a query using standard SQL. The dataFloat code converts the query to XML on the client. The XML query is transferred to the server and a result set in the form of an XML file is produced.
4. Specify a style sheet: Specify a style sheet, which will be used to render the data.
5. Transform data (render on screen): The *transform* dataFloat function takes the following arguments: metadata file name, data file name, style sheet file name. This function takes these three files and renders the data on the screen.
6. Front end functionality: Front-end functionality can be developed using standard web technology.

### **JavaScript Model**

If a client browser does not support XML/XSL, but does support JavaScript, then dataFloat will use JavaScript to enable partial client side functionality. This model is not as efficient as the XML model, but does allow some functionality to be replicated on a non-XML browser.

This model uses the same code as the XML model, however here the XML is transformed on the server into HTML. The HTML is then delivered to the client along with JavaScript code. dataFloat JavaScript code supports MS IE 4 and Netscape 4 browsers.

### **Server Side Parsing Model (HTML and WAP)**

For older browsers, the XML is parsed on the server and HTML is sent to the browser. JavaScript functionality is also performed on the server.

A similar model is used for WAP implementations. XML is parsed on the server into WML and sent to the client. As wireless devices develop in the future, dataFloat will push more of the processing to the client.

### **Palm O/S Model**

On the Palm, data structures are modeled using IBM DB2 database and renderings in C++. dataFloat generates the C++ code based on the metadata.

### **Windows Pocket PC**

dataFloat will soon support Windows Pocket PC with a full XML/XSL implementation.

## Off-Line Applications

A dataFloat database can be distributed using an off-line database model. This means that sections of the database can be downloaded to a user's hard drive for fast and reliable data access and editing without the need for a network connection. Data is updated on the user's hard drive and the central database periodically.

dataFloat off-line applications operate on an XML/XSL enabled browser. Development and maintenance of off-line applications is no different to developing on-line applications. dataFloat provides all the data transfer, storage and synchronization capabilities. This means that you can deploy and maintain off-line applications very efficiently.

Creation of mixed on-line/off-line applications is easy. Users can decide whether to use the application off or on-line, all with the same interface.

Data synchronization occurs using dataFloat compact data. dataFloat uses time stamped data records and can lock records or fields to ensure against data corruption.

### Example: Off-Line Database for B2B Applications

Take the example of a real estate agent dealing with a client. On-line systems are risky. The agent cannot risk the server or network or Internet failing while a client is waiting for a response. Nor can the agent be kept waiting for pages to download.

Off-line data distribution means that the data is stored on the agent's hard drive. Information is available very quickly and very reliably.

### Example: Email Catalogues for Off-Line Browsing

dataFloat can provide the functionality to send a catalogue (in compact format) to customers by email. The customers then have your catalogue on their hard drive, and can browse it quickly and efficiently. Updates can be provided over time, and since the off-line catalogue is browsed using a web browser, it is easy to switch on-line to order from the catalogue.

## Multilingual Functionality

dataFloat code provides multilingual functionality including functions to switch between languages, carryout multilingual searches, and display data in different languages. For measurements, dataFloat provides support for rapidly switching between different systems of measurement and currency.

dataFloat uses Unicode characters and can handle display of double byte characters on browsers which support Unicode.

dataFloat provides support for different style sheets to be used for different locales. This means that you can localize virtually all aspects of data presentation.

### 3. *dataFloat Administration Tools*

#### **dataFloat Administrator**

The dataFloat Administrator is a GUI application for development, maintenance and testing of dataFloat databases.

#### **Test Database**

The dataFloat Administrator allow for instant on-line access the database for data entry and testing.

#### **Metadata Manager**

The Metadata Manager is made up of the following sections which reflect the database structure.

Locales	Add and delete locales in the database.
Record Types	A Record Type is a collective name for data of a particular grouping.
Objects	Object Types allow for Record Types to be embedded within other Record Types. By doing this you can set up complex relationships between Record Types.
Measurements	Measurements enable the user to describe data using multi-locale measurements. e.g. multiple currencies, imperial and metric measurements.
Collections	A collection is an object that contains a set of related objects. Collection objects can be trees, vectors or graphs and are attached to record types.
Enumerations	An Enumeration is a set of values which a field may contain. Enumerations allow for storage of normalized data.
Fields	dataFloat Fields act the same way as in a normal relational database. Fields are attached to a Record Type.

#### **Refresh Files**

The dataFloat Administrator will generate new metadata XML files, new static pages and queries set up for the database.

#### **MS Front Page Style Sheet Add-In**

dataFloat includes an add-in for MS Front Page. This add-in allows you to create style sheets by dragging and dropping dataFloat data fields into the XSL.

## **dataFloat Translation Manager**

The dataFloat Translation Manager provides a user-friendly environment for human language translation. It allows for translation of all Web site content, both that which is database generated and that which is not database generated. Translation can be carried out remotely and without the need for knowledge of the database or coding behind the site. The Translation Manager can also be used with machine translation applications.

## Building Applications with dataFloat

The following explains the process for building an application using dataFloat. It shows how to build a simple search of the database and display of results. This example assumes that the system is running on MS SQL Server and IIS 5.

### 1. Initialize a New Database

The first step is to initialize a new database on the database server. To do this you run the installation script from the dataFloat Administrator. The installation script sets up the new meta tables and the stored procedures.

### 2. Set up the data structure

Using the dataFloat Administrator, you then set up the data structures. First specify the initial languages, then the record types, objects, any collections (trees), enumerations and the fields. The Administrator can be used for data entry and viewing to test the new database.

### 3. Build the query form page

To build a query page, create a new ASP file. Create a database connection object. Then call the dataFloat *getmetadata* function. The metadata file is in XML format, and a style sheet can be used to render the search form. dataFloat provides a dynamic style sheet, or by placing dataFloat functions into the XSL you can create a new one. The dataFloat add-in for MS Front Page can be used to do this.

The dataFloat *getXMLfromSQL* function is used to create an XML query, which is sent to the database for processing. dataFloat stored procedures carry out the query and produce a result set in XML format.

### 4. Style Sheet

Next a style sheet must be created to render the result set. Use a dataFloat dynamic style sheet or create one manually.

### 5. Transform Data

To render the results on the client side you call the dataFloat *transform* function and specify the metadata file, data file and style sheet. These files are downloaded and stored in the browser cache for reuse. dataFloat client side code will render the data to the screen. Other functionality can be built such as sorting, researching, reformatting using a different style sheet.

### 6. Front End Functionality

Further front-end functionality can be developed using JavaScript or other browser technology.

## Further Information

For further information, have a look at our web site: [www.idcglobal.com](http://www.idcglobal.com), or contact us in the following locations.

### USA

IDC Global, Inc  
26 Broadway, Suite 745  
New York, NY 20004  
USA  
Tel: +1 212 514 8186  
Fax: +1 212 363 9433

### Canada

2786 West 16th Avenue  
Suite 208  
Vancouver  
BC V6K 4M1  
Canada  
Tel: + 1 604 325 8861  
Fax: + 1 604 325 8852

### Europe

IDC Global (Europe) Limited  
42 Queen Anne's Gate  
London SW1H 9AP  
UK  
Tel: +44 (0)20 7808 1513  
Fax: +44 (0)20 7808 1518

### Indonesia

Suite 517C  
Gedung Pusat Niaga  
Kemayoran Jakarta 10620  
Indonesia  
Tel: + 62 21 421 8500  
Fax: + 62 21 421 8501