

12-20-00

A/RROU

Customer Number: 000959

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Attorney

Docket No. KAQ-003-1

In re the application of **Michelle Harris, Dan Jethanamest, Tom LeRoux, Corey Tripp, Peter DeLucia, Lin Hsu, Boyle Lee, Dan Liu, George Yi and Rocio Katsanis**

For: MOBILE CONTENT FRAMEWORK

Assistant Commissioner For Patents
Box Provisional Patent Application
Washington, DC 20231

CERTIFICATION UNDER 37 CFR 1.10

Date of Deposit: December 18, 2000Mailing Label Number: EL 373 209 566 US

I hereby certify that this Cover Sheet for Filing Provisional Application (37 C.F.R. §1.51(2)(i)) and the documents referred to as attached therein are being deposited with the United States Postal Service on the date indicated above in an envelope as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 and addressed to the Assistant Commissioner for Patents, Box Provisional Patent Application, Washington, D.C. 20231.

Nelson Barros

Name of Person Mailing Paper

Nelson Barros

Signature of Person Mailing Paper

COVER SHEET FOR FILING PROVISIONAL PATENT APPLICATION

Dear Sir:

The accompanying application, entitled **MOBILE CONTENT FRAMEWORK**,
is a provisional patent application under 37 C.F.R. §1.51(c) and §1.53(c).

1. ☒ The name(s) and address(es) of the inventor(s) of this application is/are as follows:

	Last Name	First Name	Middle Initial	Residence
1	Harris	Michelle		New York, NY
2	Jethanamest	Dan		New York, NY
3	LeRoux	Tom		New York, NY
4	Tripp	Corey		New York, NY
5	DeLucia	Peter		Freehold, NY
6	Hsu	Lin		Brooklyn, NY
7	Boyle	Lee		Emerson, NY
8	Liu	Dan		Washington Crossing, PA
9	Yi	George		New York, NY
10	Katsanis	Rocio		Astoria, NY

EXHIBIT

2013

2. ☐ This invention was made by an agency of the United States Government or under contract with an agency of the United States Government. The name of the U.S. Government agency and the Government contract number are:

Agency: _____
Contract No.: _____

3. ☒ The following documents are enclosed:

- ☒ 18 page(s) of specification
☒ sheet(s) of drawings
☐ page(s) of claims
☐ page(s) of power of attorney

4. ☒ Applicant claims small entity status. See 37 CFR 1.27.

5. ☐ An Assignment of the invention to _____ is enclosed.
A check in the amount of \$40.00 for recording this assignment and a recordation form cover sheet (Form PTO 1595) are also enclosed.

6. ☒ The fee for filing this provisional application, as set forth in 37 CFR 1.16(k), is \$75.00 .

- a. ☒ A check for this filing fee is enclosed.
b. ☐ Charge the filing fee to Deposit Account No. 12-0080. (A duplicate copy of this sheet is enclosed.)
c. ☐ The filing fee is not being paid at this time.

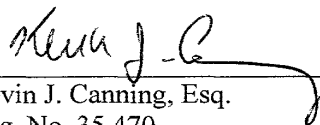
7. ☒ Please charge any fee deficiencies associated with this filing to Deposit Account No. 12-0080. A duplicate copy of this sheet is enclosed.

8. ☒ Please address all future communications to: **Customer Number: 000959** whose address is:

Lahive & Cockfield, LLP
28 State Street
Boston, MA 02109

and direct telephone calls to: Kevin J. Canning, Esq., (617) 227-7400

Respectfully submitted,


Kevin J. Canning, Esq.
Reg. No. 35,470

December 18, 2000
Date

LAHIVE & COCKFIELD, LLP
28 State Street
Boston, MA 02109
Tel. (617) 227-7400



KARGO, INC.
MOBILE CONTENT FRAMEWORK
WHITEPAPER

Case 1:19-cv-01000-Document 1-1 Filed 07/26/19 Page 4 of 20

Table of Contents

<u>Why it's Hard to Deliver Content to Mobile Devices</u>	3
<u>The Problem Today</u>	3
<u>Typical Ways of Solving this Problem</u>	3
<u>Kargo's Mobile Content Framework Solution</u>	6
<u>Kargo's Solution</u>	6
<u>Features of Kargo's Mobile Content Framework</u>	6
<u>Kargo Generic Markup Language and Translation</u>	6
<u>Application Framework</u>	8
<u>Device Identification and Capability Checking</u>	9
<u>Dynamic Image and Text Selection</u>	10
<u>Dynamic Image Scaling</u>	11
<u>HTML to KGML Translation</u>	11
<u>Page Caching</u>	11
<u>In 2Q2001: Graphical Development and Testing Environment</u>	12
<u>Architecture Overview</u>	12
<u>MCF's Core Technologies</u>	12
<u>Walkthrough of a Logical Transaction</u>	13
<u>Extensibility and Scalability</u>	15
<u>Example KGML</u>	15
<u>Integration with Other Systems</u>	16

Why it's Hard to Deliver Content to Mobile Devices

The Problem

Delivering easy to use content to mobile devices today is difficult for the content provider. There are a myriad of devices that have different functionalities and many more are expected to surface over the near term. As a result, there is a growing need to deliver this content in the most suitable and aesthetically pleasing way so that it is custom-tailored to each device, while leveraging the best features specific to every device. This delivery platform must work with today's devices, while being extensible for tomorrow's.

The following functional elements must be taken into account when delivering content to a mobile device. Figure 1 further illustrates these elements. Enterprises and content providers need an easy way to deliver the "right" content, while taking into account the below mentioned elements

- **Device language** – Devices can support languages like WML, HDML, HTML, compact-HTML, or Palm WebClippings. Even if two devices speak the same language, there is no guarantee that they each interpret and render content the same way. For example, the way WML is rendered on Nokia devices is very different from the way it is rendered on Phone.com devices.
- **Device display** – There is a large variation in the way content is displayed across different handheld devices. Some displays are long and some are wide. Some browsers can display images and some do not. Some displays support color in various depths, some support grayscale, and support only black and white. Moreover, resolution and legibility vary differently from display size. Wireless web content must be able to overcome these constraints while delivering the relevant content.
- **Device input** – Mobile phones usually only have a numeric keypad, while PDA's have stylus input. Some devices like the RIM Blackberry have a full keyboard. Some devices even support voice commands.
- **Character Encoding** – Some devices require content to be delivered using a special character encoding. For example, the i-mode phones require content in Shift-JIS encoding in order for Japanese characters to appear correctly.
- **User preferences** – Often, users will want content in a particular language or style. If a users wishes to see content in French, it should be available in French without having to author a second web site. In addition, users may prefer to see only text content in order to speed up content delivery, or prefer to see graphics-rich content on their color PDA display.

Typical Ways of Solving this Problem

Developers have tried many different approaches for solving the dilemma of serving content to all mobile devices. Here a few examples:

1. Multiple Versions of Content Approach

In this approach, the content developer creates multiple versions of web sites for each major device technology. For example, there may be one site dedicated to serving content to Phone.com-browser phones. Another site may serve content to Palm pilots. Another may serve to Nokia phones, etc. The obvious disadvantage to this approach is the issue of (a) the time and effort involved in building these sites, (b) the effort involved in keeping them updated and in synchronization, and (c) the inability to specialize content within a category (for example, all Phone.com browsers will be delivered the same content).

2. Lowest Common Denominator Approach

A content developer can build one site that delivers content to multiple devices, but only use features that are common to each device. For example, a content developer can build a wireless web site in WML, not use Phone.com extensions, and be very careful about how it is built so as to work correctly on all devices. There are numerous disadvantages to this approach: First, the site will never be able to reach every device. In this example, older wireless phones that use HDML, and even the Palm VII's, would not be able to access this site. Second, the devices that are able to interpret the site will deliver content which does not look right, and does not take full advantage of the various features of the device. In order to enable the correct display the site would have to be text-only, and difficult to navigate. For example, many phone browsers require special instructions to get the BACK button to work properly.

3. Web-Scraping Approach

In this approach, the content developer purchases a software package that takes a standard HTML web site and converts it into a language a mobile device can understand. Even though this approach might seem attractive given its low cost, most existing web sites are not suited for the content of mobile applications and the navigation of mobile devices. Moreover, maintaining a site using "web scraping" is difficult, since it requires keeping the translation rules in sync with the HTML content. Most importantly, web applications are fundamentally designed with a different paradigm than wireless web applications, and are not readily translated to mobile devices - let alone being tailored to a large set of devices. Therefore, web-scraping tools will generally not result in a wireless web site that is easy to use.

4. Common Language Approach

The best solution is to author content in any easily extended language that can be translated into a variety of different mobile device markup languages. This is how Kargo has implemented its Mobile Content Framework. Content is authored in Kargo's Generic Markup Language (KGML), and then transformed upon request to a language best suited to the end user's device. KGML is designed to enable the content developer to describe content at a more abstract level than the typical wireless markup languages like WML, HDML and HTML. These markup languages focus on how content is rendered on the device, while KGML focuses on generic, but smart, objects whose transformation is tailored to specific

devices. During this transformation process, parts of Kargo's MCF ensure the best type and length of text is used, the best type and size of image is used, and the content is well suited and customized for the device, and user.

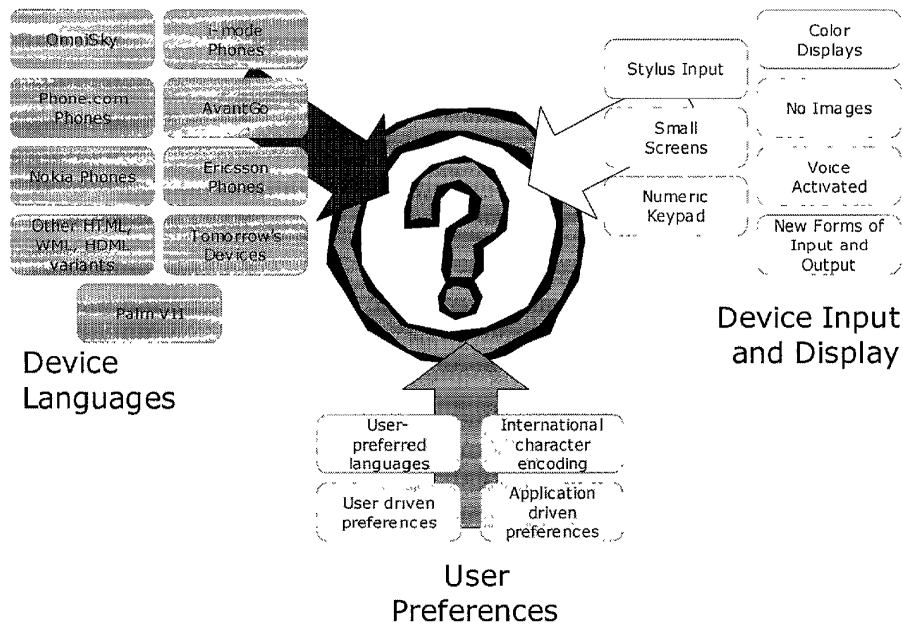


Figure 1 - Variables in Delivering Content to Mobile Devices

Kargo's Mobile Content Framework Solution

Kargo's Solution

Kargo's Mobile Content Framework (MCF) solution provides a platform comprised of many features that enable the content developer to easily build an effective wireless web site. By authoring content in KGML and using the technologies in Kargo's MCF, the content developer is able to quickly and easily build wireless web applications that target today's devices. Since each device is different, their technologies must be exploited, and their limitations overcome. Kargo's MCF handles these issues, while being truly extensible for future devices.

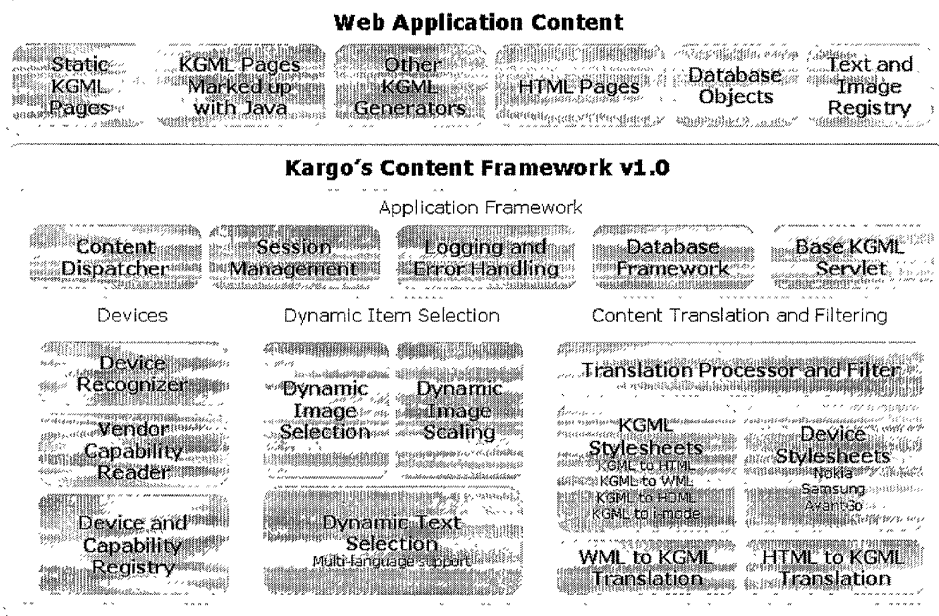


Figure 2 - Mobile Content Framework Features

Features of Kargo's Mobile Content Framework

As illustrated in Figure 2, the following features are included in Kargo's MCF. These features can be used selectively, or collectively, to provide the best content for mobile devices. Each feature is described below:

Kargo Generic Markup Language and Translation

Kargo's Mobile Content Framework offers its own brand of markup language known as Kargo Generic Markup Language (KGML) with which to describe wireless web content. This language is geared exclusively to mobile devices, and combines features of the most common markup languages like WML, HDML and HTML in order to provide the most robust features.

KGML has been carefully designed at a level above other content markup languages such as WML, HDML and HTML. KGML focuses on "logical widgets" that are translated to content that function correctly, and work better, for each and every device. KGML is designed to overcome the challenges of graphics and user-input on small devices. For example, a good paradigm for delivering content on mobile phones is a "9-grid" as illustrated in Figure 3. A graphic is displayed and broken up into 9 segments (like a tic-tac-toe board), and each number on the keypad of the mobile phone is an input link to each segment of the grid. This paradigm works well when navigating and zooming maps. This paradigm, however, does not hold true with PDA's since they do not have numeric keypads. In this case, a PDA would display the "9-grid" using a set of anchored images that can be clicked on using the PDA's stylus.

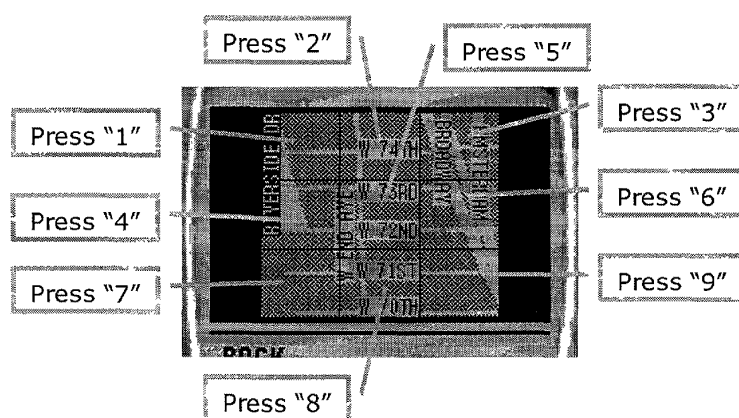


Figure 3 - Kargo's 9-Grid Technology

KGML has been developed such that it is "device-smart". It is able to determine the idiosyncrasies of each device. For example, the WAP browser on Nokia's phones do not display lists in the same fashion as the browsers on Phone.com phones. Some devices have two programmable "soft-keys", while others have one. KGML knows these limitations and features, and adjusts to them. KGML is also extensible so that as new devices are available, new translation sheets can be added easily. For example, if a new device supports a brand-new markup language, a translator for this new device can be added seamlessly, without changing any piece of the core KGML architecture.

KGML includes the following key features:

- Ability to be translated on the fly to a device's native language. Kargo's Mobile Content Framework translates KGML into other markup languages on the fly using XML-based technologies. Content is translated from KGML to languages like WML, HDML and HTML, plus device-specific variants of each of these languages. For example, some devices require their own special translation rules.
- KGML tags, which are simple to use, represent "widgets" which are at a higher level of abstraction than prior standard wireless markup languages.

For example, a KGML "Date-Chooser" renders one way on the limited screen sizes of mobile phones (possibly through a series of WML decks), and another way on a PDA (all on one screen).

- Easily extensible since new tags can be added by the content developer. The developer can specialize the behavior of any KGML tag (new and old) by updating or adding to the translation rules.
- Any system can generate KGML. KGML can originate from Java Server Pages, Active Server Pages, Perl, PHP, or many other web content management packages.

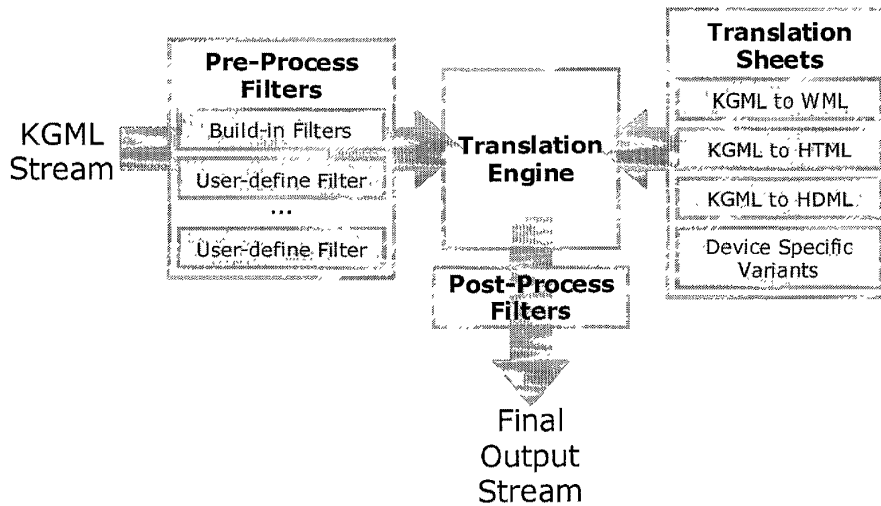


Figure 4 - KGML Translation

Figure 4 illustrates what happens when a KGML stream is fed into the translation engine. First, the stream of KGML enters into a set of filters that alter many of the "logical" tags into "physical" tags (as explained later in "Device Image and Text Selection"). Then, KGML content is translated using any number of translation sheets. The stream can then be filtered through another level of processing in order to accommodate device requirements such as character encoding.

Application Framework

Kargo's MCF framework features include:

- Logging of application messages. These messages can have levels to specify debug, informational, warning and error conditions and are important for developing and debugging large scale wireless web sites.
- Database framework including connection pooling. Database parameters are kept in a properties file to ease deployment from development to test to production.

- Automated session handling will use cookies (if the device supports them), or transparently use URL rewriting if the device does not. Session handling, including URL rewriting, is automatic, so no extra development effort is required.

Device Identification and Capability Checking

All mobile devices and their capabilities are kept in a device registry. This registry is an XML file, and allows the content developer to group devices together, and keep track of the specific features of each device. This registry also contains a set of rules used to determine which device is connecting to the Mobile Content Framework. These rules are important since all mobile devices identify themselves differently. Once the type of device is identified, attributes such as screen size, color depth, browser version and type, and translation rules become known.

Changes can be made to the device registry on an application-specific basis. In this case, however, the system allows the content developer to receive automatic updates of the device registry for new devices from Kargo while maintaining a separate registry for application specific device groupings and capabilities.

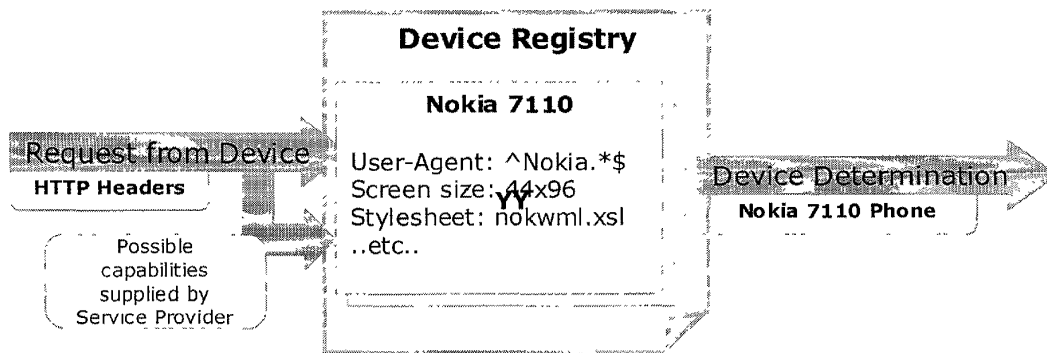


Figure 5 - Device Determination

Figure 5 illustrates how MCF determines the mobile device and its capabilities. Information is passed into the framework via the HTTP request from the mobile gateway. Some gateways identify the device and provide their own set of device capabilities, some gateways identify the device but do not provide capabilities, some gateways identify the "family" of the device, and others do not even identify the device. Kargo's MCF takes all of this information, processes it against a set of device-identification rules, and then informs the rest of the framework about the device.

Dynamic Image and Text Selection

One of the most challenging aspects of delivering content to mobile devices is delivering the right format of content. Some phones only support 1-bit BMP images, some PDA's support GIF's, and many browsers do not support images at all. The same problem arises when providing text to a mobile device. Some screens are large and some are small. For example, it may be best to limit the size of a restaurant review to one sentence when communicating with a mobile phone, but one paragraph may be better when communicating with a PDA with a much larger screen.

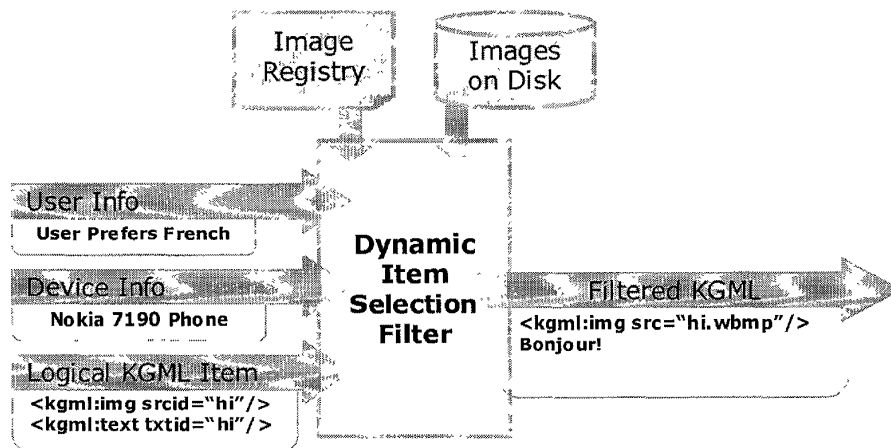


Figure 6 - Dynamic Item Selection for Images

Kargo's Mobile Content Framework solves this problem by moving the decision of which image or text element to display out of KGML, and into resource registries. The framework can use the registry to decide what is the best content to deliver to a device at any given time. This is known as "Dynamic Item Selection". Here are some of the features:

- KGML is marked up with "logical" image and text identifiers. The processor looks up the "logical" identifier in the appropriate image or text registry, and processes rules to determine which is the best "physical" item. The KGML stream is filtered on the fly to represent the "physical" item.
- Rules can be placed into the registry to deliver content based on many different variables such as the actual type of device, a device's group (devices can be placed into many different groups, such as "big browser", or "color browser", or "PDA"), and the user's preferred language (for serving different language pages without having to modify the page content!).

- Build-in filename conventions allow images to be served to specific devices, and groups of devices, just by naming the file the right way! Of course, if naming conventions are not followed, specific rules can be added to the image registry to server an image to a device.

Dynamic Image Scaling

Kargo's Mobile Content Framework includes dynamic image scaling. This feature allows a content developer to provide only a small set of images during design time, and have the framework generate images guaranteed to be displayed correctly on each device. The image-scaling framework:

- Scales and crops images to the right size
- Translates from one image format to the next
- Caches generated images to avoid translating an image more than once
- Pre-generates images to allow content developers to see all images before they are served to a device

HTML to KGML Translation

Kargo's Mobile Content Framework also includes support for web-scraping technology. This allows content to be authored in HTML, and translated on the fly to KGML, which again is transformed into content best suited for the requesting device – all without having to change the originating HTML source.

Since this technology goes through two separate translation steps, the process will always be slower than authoring in KGML alone. However, it does serve a purpose: content developers can (a) easily and quickly get an existing HTML site onto the wireless web, and (b) provide a learning bridge for building robust and full-featured applications using just KGML. In this model, content is still authored in KGML, however the KGML is embedded with rules that determine where in the HTML document to obtain dynamic content.

Page Caching

One of the biggest problems with translating content on the fly is the performance penalty paid during each web page request. Kargo's Mobile Content Framework provides multiple stages at which to cache completed page request in order to improve the performance of subsequent requests.

The page cache will work at two different levels:

- **The post-process cache** – After an application retrieves dynamic data from the database, user-preference system, or anywhere else, the KGML page will be persisted to disk. Subsequent requests for the same page will not require expensive system calls. Since this caching is performed before the device-specific translation, it is good for pages whose contents change fairly infrequently, and are not user or device dependent. For example, a page that is rendered from a fairly timely database call can be cached before filtering and translation.

- **The post-translate cache** – After KGML has been processed, the framework then translates it into a device-specific markup language (WML, HTML, etc.). At this point, the page is persisted to disk and subsequent requests for the same page do not require processing. This form of caching is good for highly customized pages that are directly targeted to a user and/or a device.

The content developer will be in complete control over how a document is cached, and how long the page remains in cache.

In 2Q2001: Graphical Development and Testing Environment

Although the current version of Kargo's Mobile Content Framework exhibits a high degree of flexibility, we recognize that it could be made even easier. The 2.0 release in early 2001 will focus on making wireless web applications easier and faster to implement. Some features that will be incorporated into this release include:

- Automated installation tool so the Mobile Content Framework will install and integrate effortlessly with pre-installed Java Servlet engines.
- Graphical development environment so content developers can build wireless applications without having to write any code. Since content may change with each device, an automated testing environment will make sure content will display correctly before they are deployed to a device. This feature will answer the question of "What content will be delivered to this device?"

Architecture Overview

MCF's Core Technologies

Kargo's Mobile Content Framework uses a few different technologies to perform its work. These technologies include:

- **Java and Java Servlets** – All components of Kargo's Mobile Content Framework are developed using the Java 1.2 SDK, and the Java Servlet 2.2 specification. You will need to provide your own Servlet engine which supports the 2.2 spec, but there are plenty of free implementations to choose from. Your choice of web server will depend on the Servlet engine being used. An example of an implementation scenario is the Apache Web Server with Apache Tomcat Servlet engine on Linux. This implementation is licensed entirely under the Apache Software License, and can therefore be deployed free of charge (please see <http://www.apache.org/> for information on these products). Keep in mind that the web server and servlet engine do not need to be the source for your web content.
- **XML and XSL** – The Extensible Markup Language (XML) is used to author KGML itself, and the Extensible Stylesheet Language (XSL) is used to author the rules that translate KGML to device-specific languages. These technologies are open standards that can be found at <http://www.w3.org/>. Kargo's Mobile Content Framework uses The Apache

Software Foundation XML Parser (Xerces) and XSL Processor (Xalan) and is distributed according to their license.

Walkthrough of a Logical Transaction

It may be a little tough to understand how Kargo's Mobile Content Framework functions without seeing the flow of processes from the beginning where a device requests a page, to the end where the framework delivers device-specific content.

Figure 7 illustrates this process.

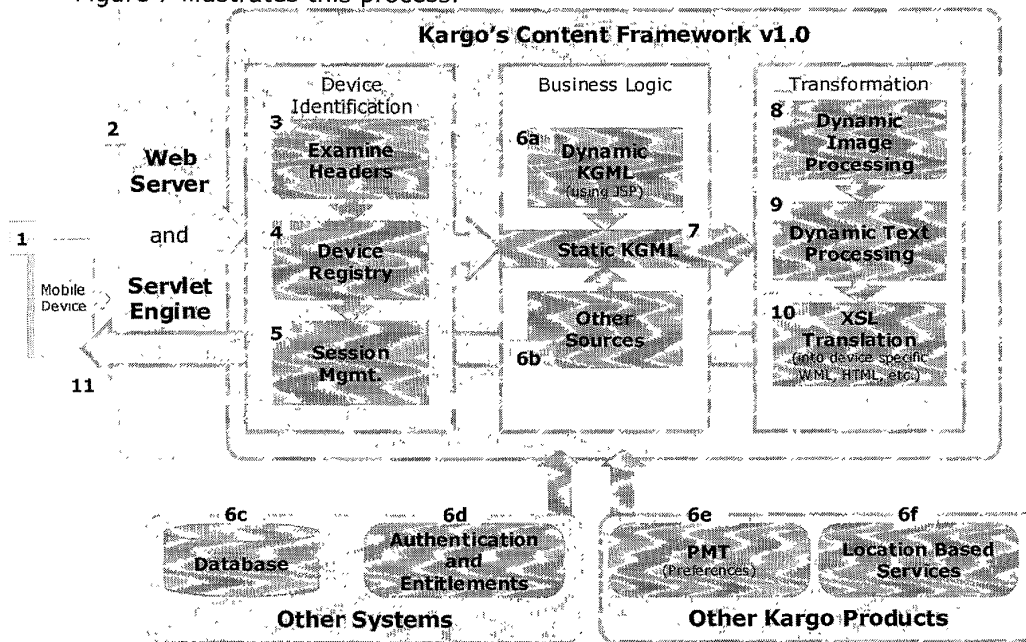


Figure 7 - Process flow of Mobile Content Framework

- 1. Device requests web page** – A mobile device or PDA requests a page from a server running Kargo's Mobile Content Framework. This request ultimately results in an HTTP page request to the Web Server.
- 2. Web server and servlet engine process the request** – The web server accepts the network connection and the HTTP request, and determines the request should be passed to the Servlet Engine. The Servlet Engine then directs the request to the appropriate page or servlet destined to generate

KGML. Please note that this configuration is entirely dependent on the web server and Servlet engine being used. However, you can be guaranteed that this process will work flawlessly as long as your servlet engine supports the Java Servlet 2.2 specification.

- 3. Device Identification** – The headers of the HTTP request are examined in order to determine the type of device and a suitable XSL stylesheet to use for the device-specific translation.
- 4. Device Registry** – An XML registry of devices is consulted to determine device capabilities (screen size, browser version, etc.)
- 5. Session Management** – A session is started to minimize subsequent processing, and serve as a temporary place for applications to store transient information (like the contents of a “shopping cart”).
- 6. Business Logic** – This is where application content is generated. Business rules, database queries, personalization engines, and other systems can be leveraged to build dynamic content for the end user.
 - a) Dynamic KGML – KGML can be generated on the fly by a Java Servlet, by a Java Server Page, or any other system which can generate dynamic content (PHP, Perl CGI, or other 3rd party content management packages)
 - b) KGML can also be generated from other sources outside of MCF.
 - c) External database can be queried using JDBC. You can also leverage the built-in database framework and connection pooling classes.
 - d) Hooks into existing authentication and authorization systems can be programmed.
 - e) Kargo’s Mobile Content Framework includes the client API to interact directly with Kargo’s personalization service so user preferences can be extracted from a global shared repository.
 - f) The API to communicate with Kargo’s Location Based Services framework is also included.
- 7. Static KGML** – Once all dynamic “business” processing is complete, the resulting KGML is fed into the content translation system.
- 8. Dynamic Image Processing** – The KGML stream is examined in order to substitute the most appropriate image with regards to the device, and user preferences.

9. Dynamic Text Processing – The KGML stream is examined in order to substitute the most appropriate text with regards to the device, and user preferences.

10. XSL Processing – KGML is translated into a language the requesting device can understand. The stylesheet determined in Step 3 is used to perform the translation.

11. Content is sent back to the device – The requesting device obtains the content, and renders it for the user.

Extensibility and Scalability

Kargo's Mobile Content Framework is easily extendible and scalable. The content developer can configure and add to the framework in the following areas:

- Tags can be added and removed from the KGML language. A developer can include new KGML tags that can almost anything. A new tag can be a composite of existing tags, or it can represent an entirely new idea.
- Content filters can be inserted to add new functionality, or address special characteristics of new types of devices (special character encodings, scripting languages, etc.)
- As new devices become available, the content developer can freely add these new devices to the registry, and rules to recognize those devices. Alternatively, a subscription plan can ensure that an enterprise always obtains the most recent version of this file.
- Database classes allow many different options for pooling database connections in order to perform fast database access.

Example KGML

The following is a sample of KGML which illustrates some of its basic features.

```
<?xml version="1.0" encoding="utf-8"?>
<kgml:kgml xmlns:kgml="http://www.kargo.com/kgml" version="0.9">
  <kgml:doc version="1.0">
    <kgml:title>NY Nightlife</kgml:title>
    <kgml:block id="splash">

      <!-- EXAMPLE 1 -->
      <kgml:button href="index?a=main"
        keytype="accept" type="go" labelid="enter"/>
      <kgml:p align="center">

        <!-- EXAMPLE 2 -->
        <kgml:img srcid="logo" alt="My Nightlife"
          border="0"/>

    </kgml:block>
  </kgml:doc>
</kgml:kgml>
```

```

<!-- EXAMPLE 3 -->
<kgml:text id="welcome"/>

<!-- EXAMPLE 4 -->
<% if ((String)session.getAttribute("lang") ==
    null)
    out.println("<kgml:a
    href=\"index?a=chlang\">Choose
    Language</kgml:a>");
%>
<kgml:br/>
<kgml:br/>
</kgml:p>
</kgml:block>
</kgml:doc>
</kgml:kgml>

```

Example 1 shows the abstract nature of KGML. The `<kgml:button>` element will be displayed as a "soft-key" for WAP devices, but as a "link" for devices which understand only HTML.

Examples 2 and 3 illustrates KGML's Dynamic Item Processing feature. Notice that in Example 1 the text "logo" is placed into the `srcid` attribute instead of the URL to the actual image. The text "logo" is used to index into a set of rules to determine the best image to display for the specific device. For example, the framework may determine that "logo.gif" is the best image for the given device, or possibly "logo-nok7100.wbmp" is the best image. The important thing to understand is that the KGML itself does not determine the "physical image" to display – only the "logical" image.

Example 3 illustrates the same principal, but with text. The id "welcome" may be translated to "Bonjour" for users who specify French as their preferred language, or possibly "Welcome to our site, please take a look around!" for wireless devices having a larger display.

Example 4 shows how Java code can be inserted directly into the KGML. In this case, a Java Server Page is rendering the KGML. However, remember that KGML can be generated by many different web technologies like ASP or Perl CGI.

Integration with Other Systems

Kargo knows that content can and will be generated from a variety of different web systems. Although Kargo's MCF uses Java Servlets to perform its work, content can be generated using any web technology.

- **KGML Content** – Almost all web technology tools can be leveraged to generate KGML. Since KGML is just a plain XML document, tools like PHP, ColdFusion, Perl CGI, Vignette, Microsoft Active Server Pages, and Java Server Pages can be utilized to generate dynamic KGML. Kargo's MCF dispatcher will automatically query these services, and translate the content to a form best suited to the requesting device.

- **HTML Content** – Using Kargo’s HTML to KGML engine, content can be written in HTML and converted on-the-fly to a language best suited to the requesting device, without changing the source HTML.
- **XML Content** – Content developers can generate content using XML, and then provide XSL stylesheets to translate the XML into *any* markup language. This is ideal in cases where dynamic XML content is obtained from a database, or third-party source, and needs to be rendered into KGML, or any other XML variant.

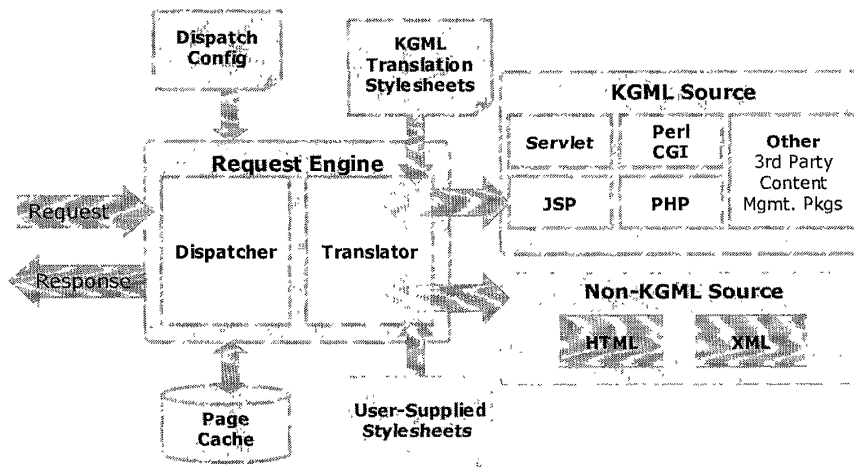


Figure 8 - Integration with Other Content Systems

Figure 8 illustrates how content can originate from different sources, and be translated on the fly by Kargo’s Mobile Content Framework.

- 1. The wireless language first requests a page from the MCF.** The framework will immediately determine the type of device with which it is communicating, and initiate a session.
- 2. The dispatcher picks up the request, and determines from where to obtain the data.** The dispatcher can choose to serve the page from cache, or obtain the page again from the data source.
- 3. The data is retrieved, and translated.** The appropriate data source is contacted, and the resulting stream of data is obtained. This data can be KGML, or any XML file. It can even be plain HTML. The data source can be a web site, a static file, a Servlet or a Java Server Page. Once the data is

1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2

- 2000-11-09-CF-1