



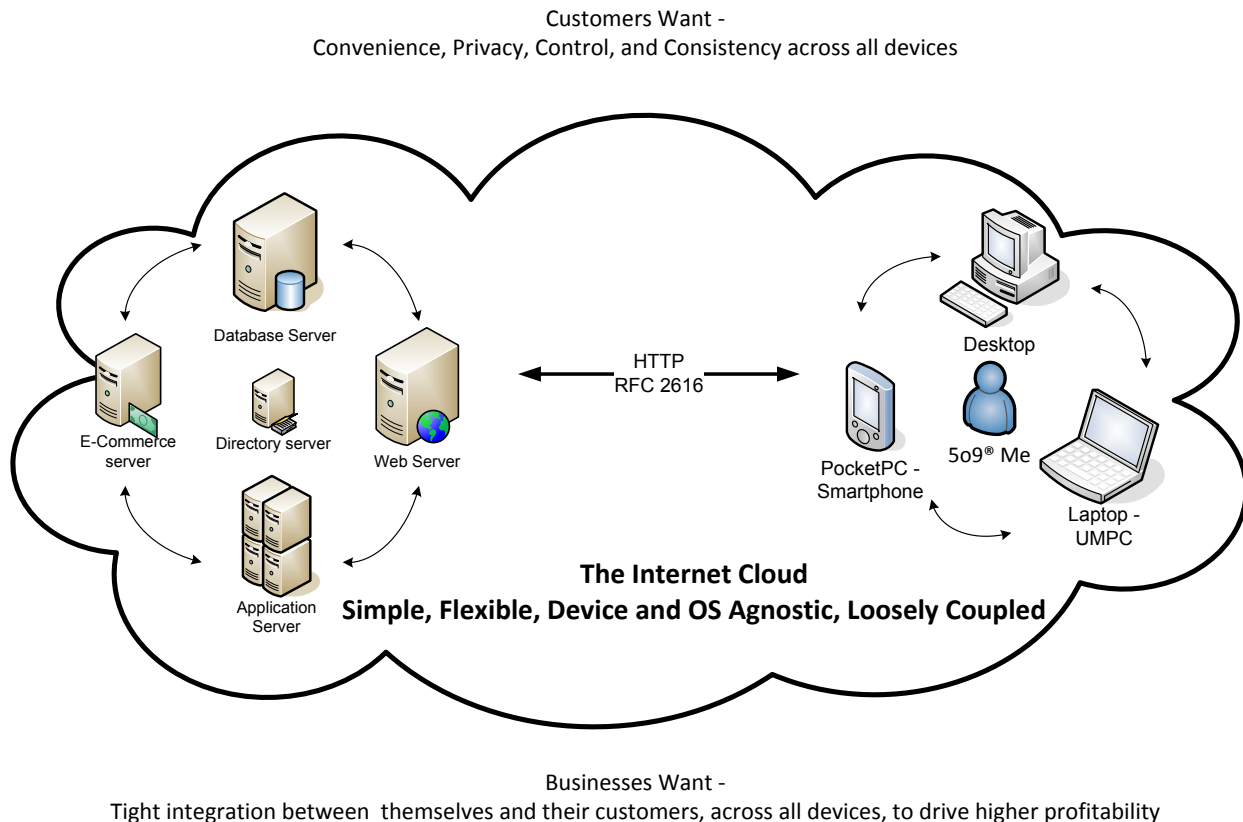
## Delivering Increased Relevance with (Mobile) Meta Data



Determining Dev Cap  
Using the HTTP Protocol

## There is only one Internet.

It works because it's simple, flexible and works on all platforms. It also works because it's loosely coupled - there is no need to upgrade the browser every time the Web publisher changes a site or adds to its service offerings. For the past decade this model has been effective with users allowing them to enjoy access to the Internet via their browser on desktop and laptop PC's with large screens, full sized keyboards and an external mouse. However, the world has changed and new "screens" (TV and Mobile) have emerged that can access the Internet from anywhere and this has created a new set of challenges.



**Diagram 1**

## Confronting Reality

So how do we extend the Web to alternative screen users given the different capabilities of all of these devices?

- Small, inadequate keyboards that restrict data entry required for adding additional context
- No easy programmable approach to access any device data via the browser
- Small screens that restrict what may be effectively displayed
- No mouse for quick & easy pointing & clicking

Up until now developers have pursued two primary approaches:

1. Build standalone, customized applications which do not require a browser to access the Internet.
2. Segment the Internet into multiple versions – for example the Desktop, the Mobile and the TV Web.

Both approaches are time consuming, expensive for the Enterprise, and require the user to learn something new. A new approach should be adopted that leverages the following:

1. One platform – the Web
2. One interface – the Browser
3. Access to multiple data sets – the Context

This results in a consistent experience that scales across the three “screens of the future”. The PC, the TV and the MC (Mobile computer)

## The Problem

So what is the underlying technical challenge to achieving the above goal of one platform, one interface and access to multiple data sets? It can be summed up in two words – Device Capabilities (colloquially known as DevCap). Currently it’s very difficult to determine in real time, the exact capabilities of the device that is connecting to the Web server. For example – does the device have a bio-fingerprint reader, what exactly is the browser window height, what is the processor id of the device?

While current browsers do transmit some information, invariably it’s not always correct and is certainly not detailed enough to determine the exact device capabilities. Therefore the Web admin/programmer has to design to the lowest common denominator when delivering a Web page. Usually this results in a poor user experience. How many times have you clicked on a link in a Mobile browser only to have it fail to correctly display? It’s a simple test, yet it shows how much room there is for improvement.

### A Standards-Based Approach to Solving the “DevCap” Problem

Over the years multiple approaches to solving the DevCap problem have emerged, from asking manufacturers to add more device data to the browser user agent, to online databases that are compiled for different device capabilities, to HTML5 and the introduction of the [W3C Geolocation API](#) (application programming interface). The API is agnostic of the underlying location information sources, including Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, Wi-Fi and Bluetooth MAC addresses, and GSM/CDMA cell IDs, as well as user input. There’s only one problem – no guarantee is given that the API returns the device’s actual location!

In addition, the API doesn’t support new capabilities that will be added to future devices. The result is a standard that lags behind – not ahead of innovation. It’s time to look at a different way to solve the problem. One that will scale to all devices connecting to the Web (see diagram 1), leverage all the existing standards, and stay ahead of the innovation curve.

## The Solution

The common denominator in all of this is the HTTP protocol. It in effect, binds every Web-enabled device to the multiple data sets that can drive more relevance “if” they have access to more context about the user, the device and the location from which they are connecting. At its core, the Internet is a communication medium that is governed by [RFC 2616](#) which defines the protocol as follows:

*The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, **stateless**, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through **extension of its request methods**, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.*

In simple terms the HTTP protocol is a request/response standard between a client (a browser) and a Web server. It’s a “stateless” protocol, which means that the server treats each request as an independent transaction that is unrelated to any previous request. This approach simplifies server design because there is no need to dynamically allocate storage to deal with the conversations in progress. The disadvantage is that it may be necessary to include additional information in every request. The server must interpret this additional information, increasing the server load and the algorithm complexity required to generate request responses.

However with current Web servers being more than capable of handling complex algorithms it should be possible to simply add (via standard programming techniques) additional small pieces of Meta Data to the outgoing HTTP request header. (A header defines various characteristics of the data that is requested or the data that has been provided – think of this as “Meta-Data” or Data about the Data.)

The HTTP protocol, through extension of its request methods, provides for a standard way to add new “non-standard” data. It’s called the X Header. X-headers are fields in the HTTP request header beginning with an X (e.g. HTTP\_X). They can be used for many different things: User identification, device recognition, probing for network characteristics, real time location information and much more. With the ability of the protocol to support new X headers all that is now required is a way to actually add them dynamically as a request is leaving the device. For this you can use a standard MIME filter. A MIME filter is an asynchronous (two way) pluggable protocol that receives data through a stream, performs some operation on the data then returns it to the data stream. Nearly all modern browsers support pluggable MIME filters.

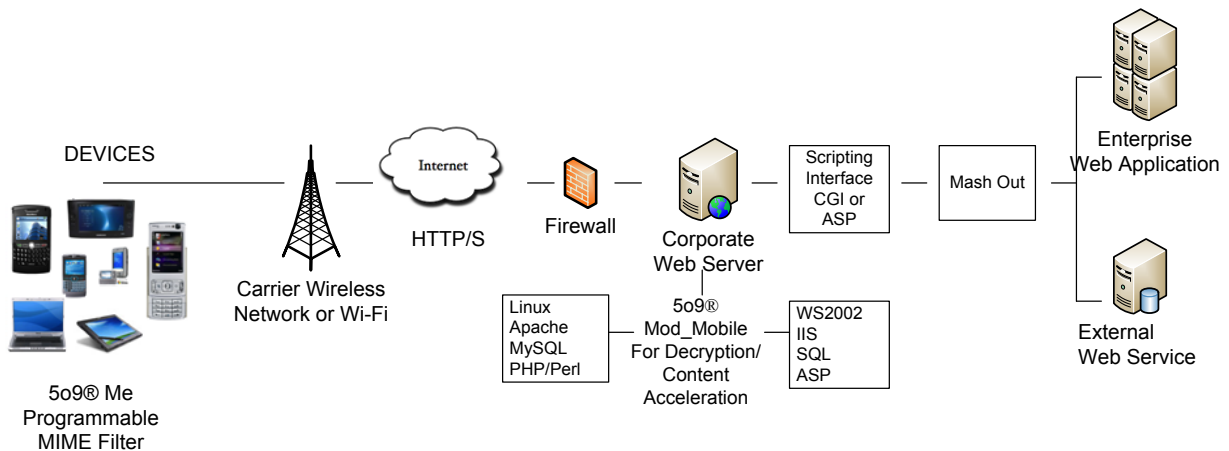
Now it is possible to add a filter to the browser that “talks” to a Mobile application running on the target device. The purpose of this mobile application is to collect the device’s Meta Data. It can literally be **any** data that is accessible via the standard operating system API’s or that can be entered by a mobile user. This data is usually broken down into three categories:

1. User Information – Name, Address, Phone number, Personal Preferences
2. Device Information – Carrier network, Screen Colors, Screen width
3. Location Information – Latitude, Longitude, Altitude, Speed, Direction, Cell Tower ID

There is virtually no limit to the data that can be stored in this Meta Database. In addition it can be encrypted to enhance user privacy. With this last item in place it is now possible to add new Meta Data in real time to the existing HTTP protocol using all the current standards. The steps are as follows:

1. The Meta Data is stored within the lightweight Mobile application
2. The consumer opens the browser and navigates to a Web address
3. This triggers the MIME filter to request information from the Meta Database
4. This data is then added to the outgoing HTTP Request headers
5. The new data is sent in the following format, e.g. : HTTP\_X\_LATITUDE="37.474105"
6. The data arrives at the Web server (where it is decrypted if necessary)
7. The Web server uses a simple script to read the incoming headers and parse the required data before sending it to any Web application

Schematically it looks like this:



**Diagram 2**

## Data Privacy

One of the main concerns for businesses and consumers is the ability to control data privacy and what data gets shared with whom. Using the outlined design, this is achieved in a straightforward manner. The Mobile application that contains the user's Meta Data allows for individual field control over each piece of data. Checking or un-checking the data fields transmits or stops data transmission. Data may be grouped or organized to support things such as home and away data sharing profiles and approved data sharing domains, or white lists, and may be locked down or changeable based upon the privacy and security preferences of consumers and businesses.

## Relevance Drives Results

The core technical problem that has emerged with the advent of new Internet-connected devices is the need to access in real time, the exact device capabilities of the connecting device. 5o9's approach leverages the one protocol that joins all of these devices together in the loosely-coupled design that makes the Web so powerful. This approach also provides a programmable solution for the developer so they can add whatever data is necessary to support their business and that results in a better experience for their users.

Now that the Web server can receive additional data in each request, it is possible to use this information to deliver more relevant results to the mobile user making the Web request. Context drives relevancy – the higher the relevancy the greater the ability to monetize that outgoing Web response.

- Increase productivity by minimizing data entry
- Increase loyalty by delivering a customized user experience
- Reduce transactional friction with location-specific or account-specific responses
- Decrease abandonment rates by only sending device-appropriate content and reducing page load times

## About 3PMobile™

3PMobile™ is a software company that enables enterprises to manage the 3Ps of the mobile Web experience – performance, privacy and personalization. To get the most from your mobile Web strategy, you need to know in real-time, who the user is, what their device is capable of and where they are. That's what 3PMobile™ software does. It uses the browser to deliver real-time who, what and where data to your Web server for use in any Web app. Access to this data is the foundation for optimizing mobile Web performance, managing privacy policies and delivering a compelling and personal mobile Web experience.

To learn more about how to enable your Mobile SaaS strategy, please visit us at [www.3pmobile.com](http://www.3pmobile.com) or call (303) 938-1769.

