

# PageBox

## Executive summary

I present here a new approach for software deployment whose tentative name is PageBox. It leverages on existing Java standards (Web Archives, Java Server pages, servlets) and technologies, especially:

?? Class Loaders

?? Sandboxes and Java 2 security

PageBox aims at

?? Providing sub-second response time to browser based applications

?? Reducing bandwidth need for ISP infrastructure

Its core concept is to allow Application Servers to handle Web Archives like browsers handle applets. It is implemented today as a servlet package that can run in free (Tomcat) or inexpensive (Resin) Application Servers and run on a large range of devices. It could be integrated in these products. It could also be embedded in appliances.

It is designed to be operated by ISPs. It conforms to Internet rules, no central administration and almost unlimited scalability through the use of well-defined protocols.

Though this approach implies deploying PageBox and Web Archives on a large number of computers, I show PageBox can be securely administrated and efficiently troubleshot. I also pay special attention to address security issues: Web Archives can be published only by identified entities and network traffic can be protected against tampering and eavesdropping.

Its standardization should interest companies with remote locations, customers and partners that need a sub-second response time and would like to benefit of Web applications advantages, shortened time to market, simplicity and low development cost.

It should also allow ISPs that host Web Applications to increase their revenues and software companies to develop a large range of new applications. Vendors could also sell PageBox appliances.

# Table of content

<b>1</b>	<b>PROBLEM STATEMENT</b>	<b>5</b>
1.1	Web application	5
1.2	Graphical front end	5
1.3	A third way	5
<b>2</b>	<b>SOLUTION BASED ON JAVA AND J2EE</b>	<b>7</b>
2.1	Implementation	7
2.2	Administration	8
2.3	Security	8
2.4	Analysis	9
<b>3</b>	<b>ISP SOLUTION</b>	<b>10</b>
3.1	Principles	10
3.2	Actors	10
3.3	Analysis	11
3.3.1	Web Caching	11
3.3.2	PageBox integration	12
3.3.2.1	Principle	12
3.3.2.2	Session handling	13
3.3.3	Protocols and security	14
3.3.3.1	Client/server protocol	14
3.3.3.2	End user security	14
3.3.4	Archive publication and distribution	15
3.3.4.1	Archive distribution	15
3.3.4.2	Archive publication	16
3.3.4.3	Charging model	17
3.3.4.4	Legal aspects	17
3.3.4.5	Archive transformation	18
3.3.5	Reference data	18
3.3.5.1	Serialized objects	18
3.3.5.2	JMS	18
3.3.6	Local data update	19
3.3.7	Life cycle	19
3.3.8	Troubleshooting	20
3.3.9	PageBox API	22
3.4	Advantage analysis	23
3.4.1	Traffic	23
3.4.2	DSL and Cable network	25

PageBox	Page 4
3.4.3 Markets	25
<b>4 STANDARD NEED</b>	<b>26</b>
4.1 PageBox	26
4.2 ICP	26
4.3 Publication protocol	27
4.4 Summary	28
<b>5 AUTHOR BIOGRAPHY</b>	<b>29</b>

## Table of Figures

FIGURE 1: INTRANET SOLUTION	6
FIGURE 2: CLASS DIAGRAM	7
FIGURE 3: ADMINISTRATION	8
FIGURE 4: ACTORS	10
FIGURE 5: WEB CACHING	11
FIGURE 6: AREAS	12
FIGURE 7: CLIENT/SERVER SECURITY	14
FIGURE 8: PAGEBOX DISTRIBUTION	15
FIGURE 9: LIFE CYCLE	19
FIGURE 10: PAGEBOX LOG DISPLAY	21
FIGURE 11: PAGEBOX STATISTICS	22
FIGURE 12: PROTOCOL COMPARISON	24
FIGURE 13: MULTIPLE ISP DEPLOYMENT	27

# 1 Problem statement

Today to offer a Graphical User Interface a company must either:

1. Write a Web application or
2. Write a graphical front end

## 1.1 Web application

Advantages:

- ?? A Web application is easier to write and to maintain than a graphical front end. It also requires less skill.
- ?? A Web application is a central application, so it is easy to deploy and update.

Drawbacks:

- ?? A Web application being a central application also means all application parts, presentation, business logic, data caching and accesses run on a small set of servers. Large server resources (memory, CPU and disks) are more expensive than small computers ones
- ?? Browsers are used to display Web application pages and these pages are downloaded using HTML or XML over HTTP. Here the main drawback is that presentation is downloaded with data. As a consequence Web applications require more bandwidth than applications invoked by graphical front ends

Web Applications are successful and address well End Consumer market where availability and response time requirements are lower. The End consumer doesn't pay nor is paid to use the application but I think the major point here is she or he is an occasional user. Compared to a Professional User, she or he is still a beginner and therefore slower.

## 1.2 Graphical front end

Advantages:

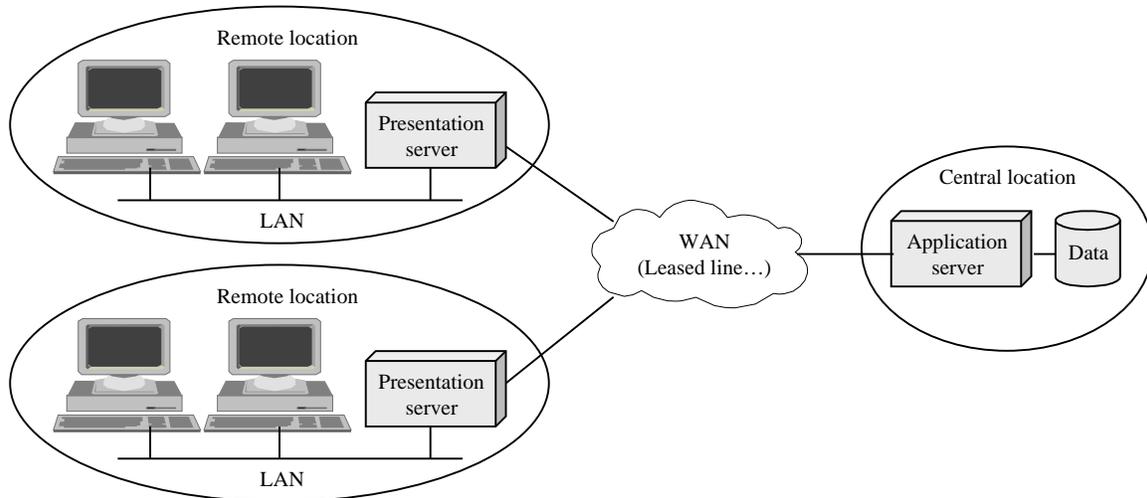
- ?? From a communication point of view, a graphical front end is the client part of a client/server application. It can use client/server protocols such as EJB over RMI/IIOP, which carry only data and require less bandwidth than Web applications
- ?? It runs presentation on the client and requires less resources on server where they are expensive

Drawbacks:

- ?? A graphical front end is harder to develop and to maintain. It is more demanding in project management and developer skills. This complexity doesn't accommodate time to market and frequent changes constraints
- ?? A graphical front end is hard and expensive to deploy and update on a large number of devices

## 1.3 A third way

Both solutions are not very satisfactory for Professional Users.  
A third solution has been successfully deployed on Intranet.



**Figure 1: Intranet solution**

Here the Web Application is split in a presentation part and an application (business logic + data access) part. The presentation part is deployed in every site. The application part remains on the central site. The presentation part calls the application part using a client/server protocol such as EJB over RMI/IIOP.

This solution combines advantages of both Web Applications and Graphical Front End:

- ?? It has the development simplicity of a Web Application
- ?? It uses HTML/XML over HTTP only on remote location LANs. It uses a client/server protocol on WAN with minimal bandwidth requirement
- ?? It spares resources on the application server where resources are expensive. A presentation server handles only the users in the same Remote location and can be hosted on an inexpensive machine of the same type as users' workstation
- ?? It simplifies deployment. Only presentation servers have to be deployed and maintained

The solution has two drawbacks:

1. It is only suitable for Intranet. Presentation servers supervision and maintenance requires manpower, skills and raises security issues
2. It is not cost-efficient when Remote location contains less than five or ten workstations

## 2 Solution based on Java and J2EE

### 2.1 Implementation

I developed under Gnu Public License 2, a simple implementation addressing the first shortcoming of the third solution.

Its principle is to allow the presentation server to act as a browser and download presentation archives as a browser downloads applets.

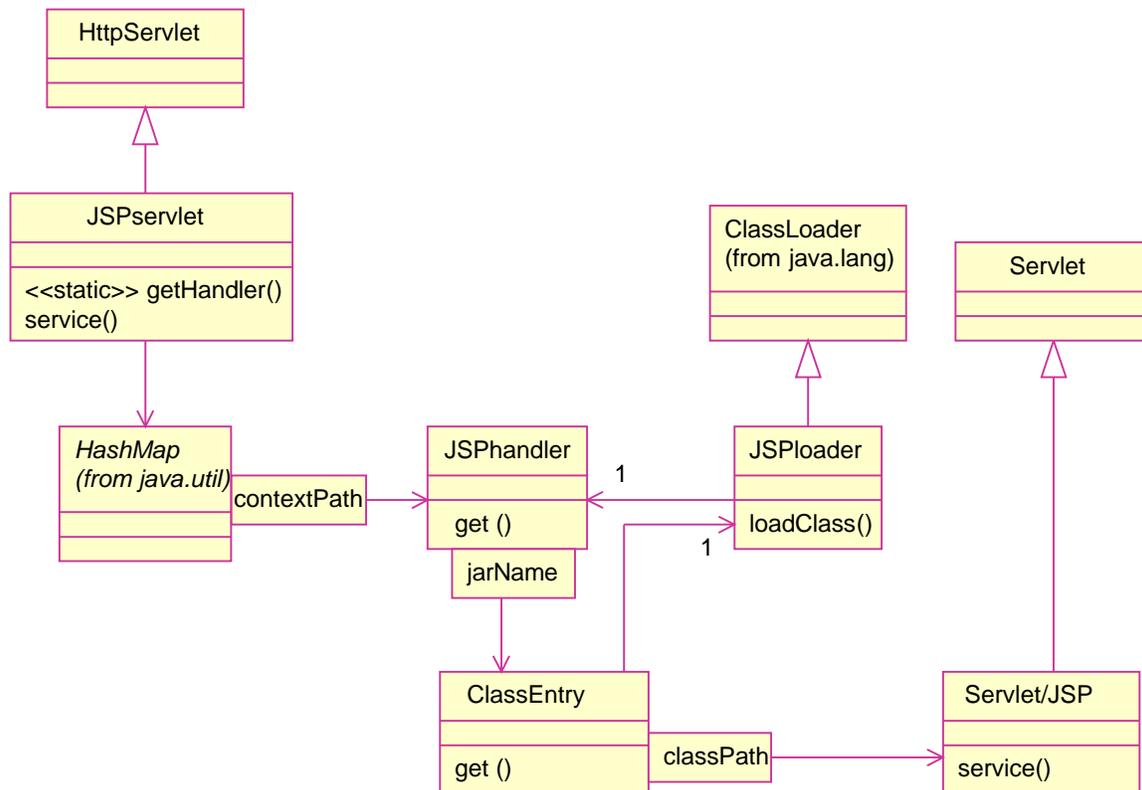


Figure 2: class diagram

It contains a service servlet JSPervlet, invoked by the servlet container. Depending on the path, it looks for an archive. If it doesn't find it creates a class loader JSPloder, which downloads the archive from a remote server. Then a ClassEntry class instantiates the requested servlet or JSP using the created class loader.

PageBox is packaged in a war file, whose init-parameters specify parameters such as the Certificate Authority and Certificate Revocation List URL.

## 2.2 Administration

A servlet allows administrating PageBox.

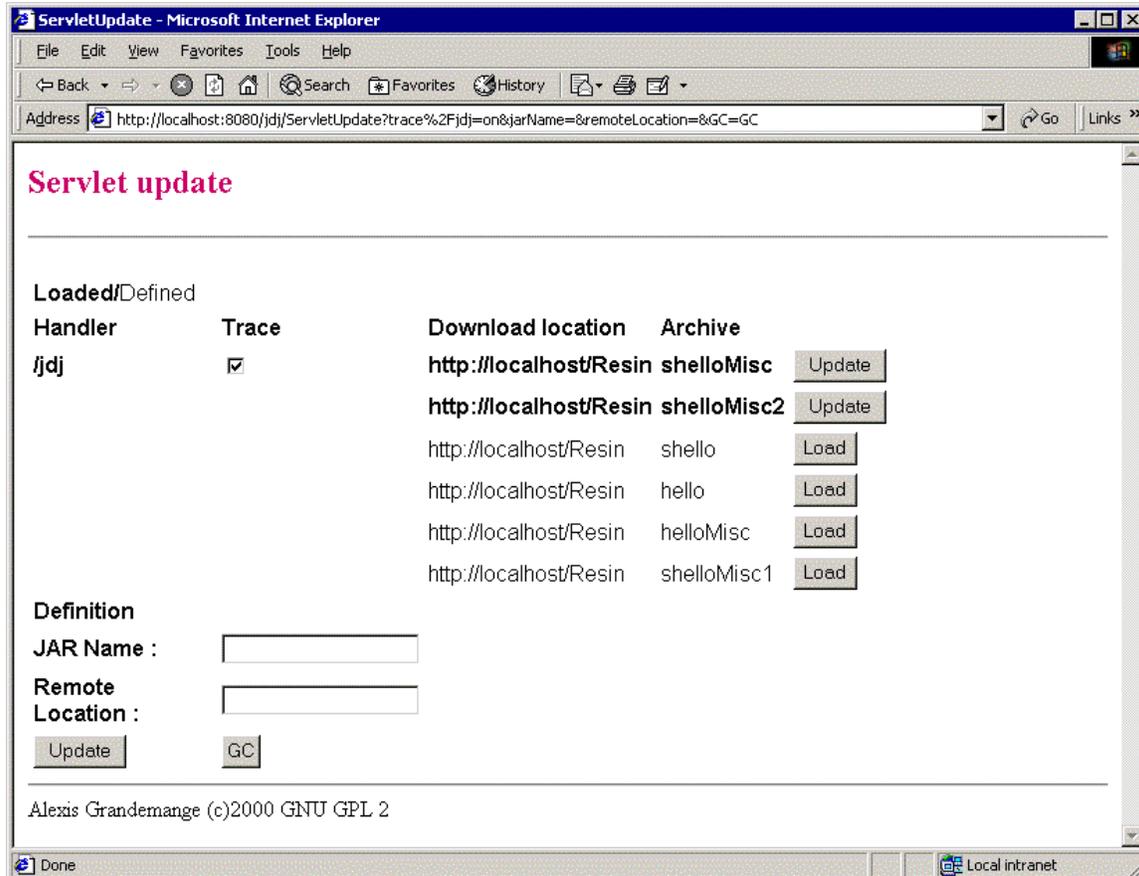


Figure 3: administration

This servlet allows:

- ?? Adding or changing managed archives at the bottom of the screen
- ?? Listing managed archives and triggering their download or update

The servlet uses GET mode, so it is easy to issue administrative requests with batch commands.

## 2.3 Security

PageBox can use JSSE to download archives using SSL.

It also support signed archives and security using the Sun JKS key store and policy files.

When JSPloder loads an archive class:

- ?? It retrieves the certificate chain the class has been signed with
- ?? It checks the validity of the certificate chain with a Certificate Authority
- ?? It checks the certificate has not been revoked
- ?? It defines the class in a protection domain whose permissions are the permissions associated with this certificate and code source

When the class is instantiated, it runs in a sandbox and is only allowed permission it was granted in the policy file.

---

## 2.4 Analysis

PageBox is a reasonable technical answer to the problem. It is:

1. A simpler way to manage Intranet applications
2. An appealing solution for B2B communication: suppose company A has written a Web Application. It hosts the business logic and data access part on its site. Its customer, company B downloads the presentation part, which acts as a smart proxy for its internal users. It is a win-win situation: Company A runs its application on a smaller farm and company B get a better response time thank to a smaller bandwidth requirement.

But it fails to fully address shortcomings of third way.

- ?? It still requires the installation of a Java Server on each remote location
- ?? It is still unable to address efficiently smaller remote location needs (one to three workstations)

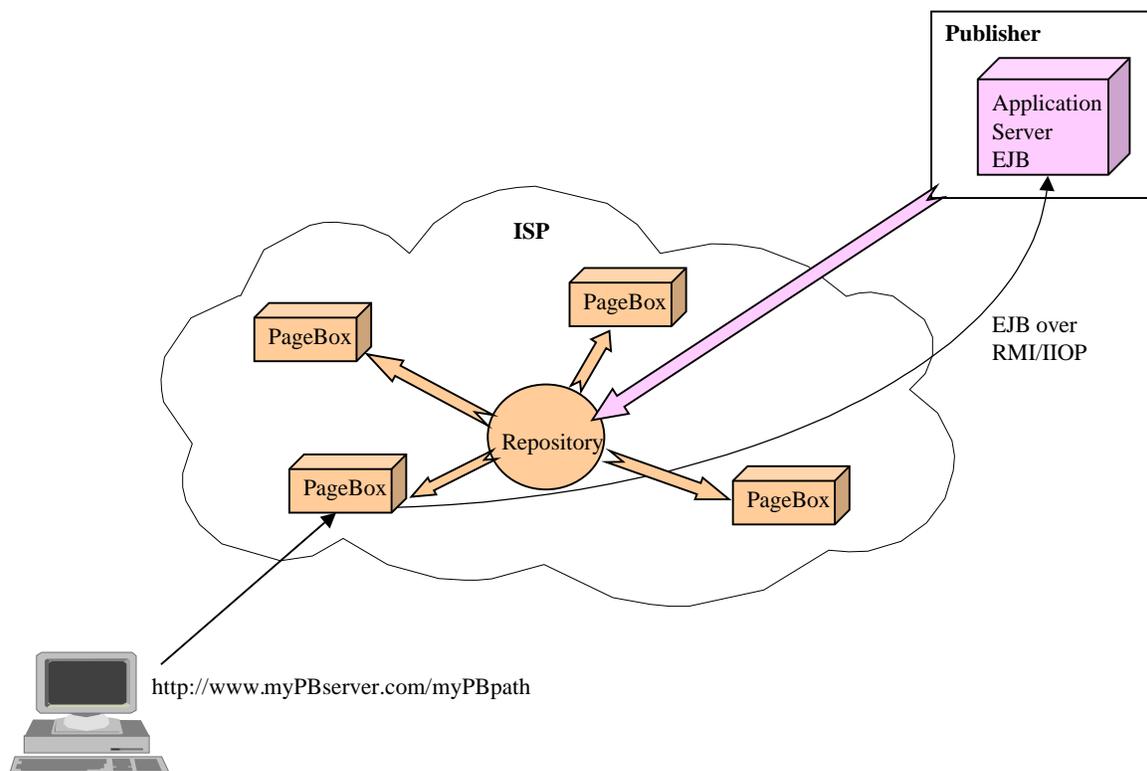
The problem is a software solution deployed on the Internet end points cannot be optimal in term of resource use. The only way to address this issue is to ask ISPs to host PageBox.

## 3 ISP solution

### 3.1 Principles

- ?? PageBox is a new service the ISP charges to the publisher
- ?? Presentation (Web Archive) hosting is a commodity like routing or proxies
- ?? PageBox instances host Web Archives from different publishers
- ?? ISPs host PageBox instances where it is needed on their network. Archives deployment on these PageBox instances can depend on archive use. A highly used archive will be deployed on more PageBox instances than a less used archive
- ?? PageBox uses existing standards, especially Web Archives and Java 2 Security

### 3.2 Actors



**Figure 4: Actors**

- ?? Web user. She or he invokes an application using an URL
- ?? Publisher
  - It hosts the business logic and data access in its Application server
  - It publishes its presentation (Web Archive) to the ISP defined entry point
- ?? ISP
  - It deploys PageBoxes on appropriate locations
  - It maintains a repository cluster that acts as an entry point for publishers and distributes updated archives to PageBoxes
  - It hides PageBoxes to the user: a user must be able to use the same URL, whatever PageBox is invoked

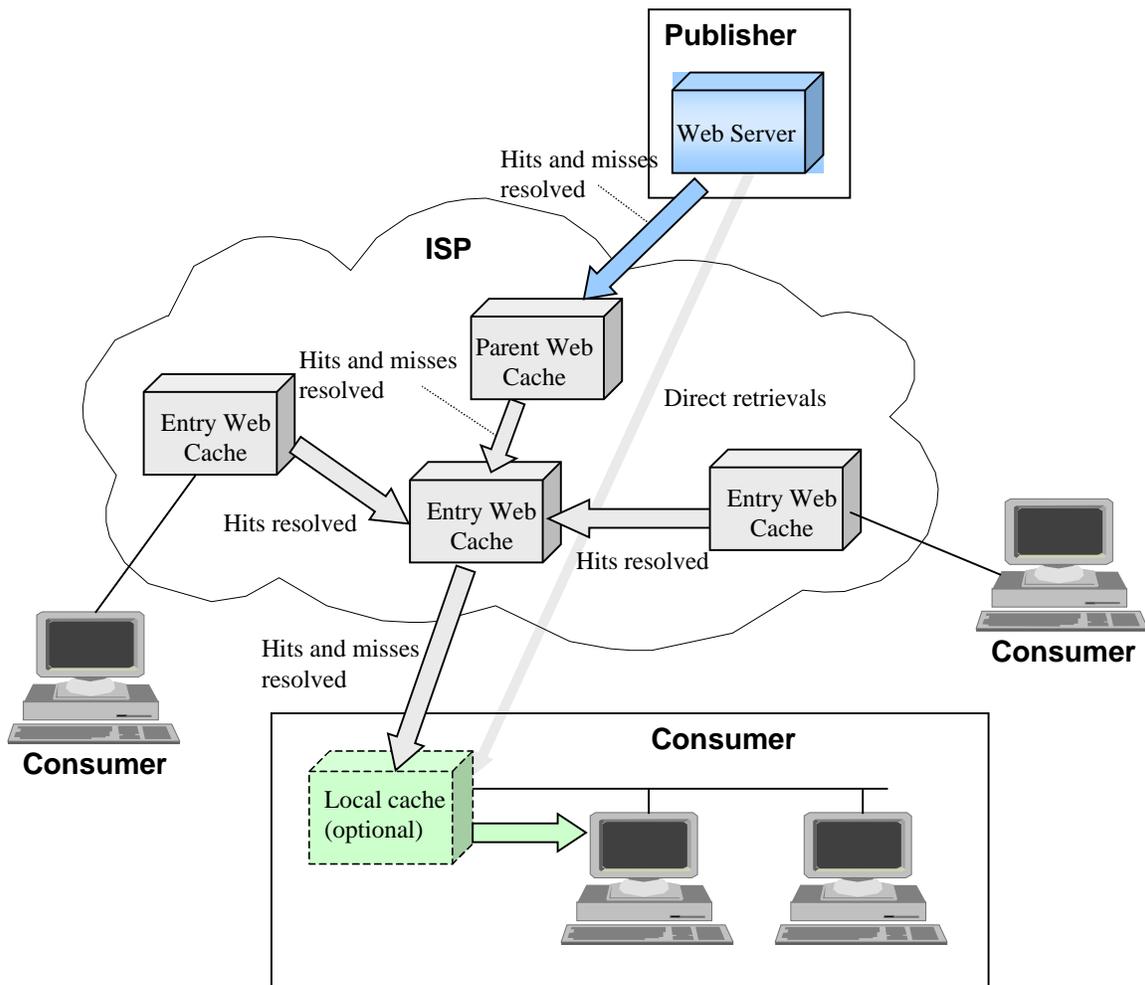
### 3.3 Analysis

Figure 4 arrows show the different issues that the ISP solution must address:

- 1) A browser at any location must be able to invoke PageBoxed presentations with the same URL. The address part www.PBserver.com on Figure 1 must be the existing address of the publisher
- 2) PageBoxes must be able to connect to publishers using protocols like EJB over RMI/IIOP in a secured mode, IPsec (VPN)
- 3) The publisher must publish archives to the ISP repository
- 4) The ISP repository must distribute archives to PageBoxes

#### 3.3.1 Web Caching

PageBoxes will act with dynamic content like Web Caches act with static content.



**Figure 5: Web caching**

The ISP deploys one or many Entry Web caches and upper layer parent cache(s). Entry Web Caches are neighbors. For an Entry Web cache other Entry Web Caches act as sibling Web caches.

When a browser doesn't hold a page, it asks it to a local cache (if one is defined) or to the ISP Entry cache.

If the local cache doesn't hold a page, it asks it to the closest ISP Entry cache. If it doesn't hold the page the ISP Entry cache asks its neighbors if they have the page in cache using a standard protocol, ICP defined by RFC 2186 and RFC 2187. If no neighbor has the page, the cache forwards the request to the parent cache.

If the parent cache doesn't hold the page it retrieves it from the provider site. Direct retrievals happen when the content cannot be cached.

### 3.3.2 PageBox integration

#### 3.3.2.1 Principle

The ISP divides its network in logical areas.

An area must contain:

- ?? Two or more PageBoxes for fault tolerance
- ?? Two or more Web Caches

An PageBox-enabled Web Cache processes cacheable pages as described on Web Caching section. But if the page is not cacheable, instead issuing a direct retrieval it looks in a table if the URL is handled by neighbor or upper layer PageBoxes.

If neighbors PageBoxes can handle the URL, it select one of them with a round robin algorithm, otherwise it selects an upper layer PageBox. If no PageBox can handle the request, it issues a direct retrieval.

This table can be build and updated by multicasting standard ICP messages ICP\_OP\_QUERY to PageBoxes. They answer ICP\_OP\_HIT if they can handle the URL or ICP\_OP\_MISS if they cannot.

The ISP can operate PageBoxes at different levels, Region, sub-region, area on Figure 7.

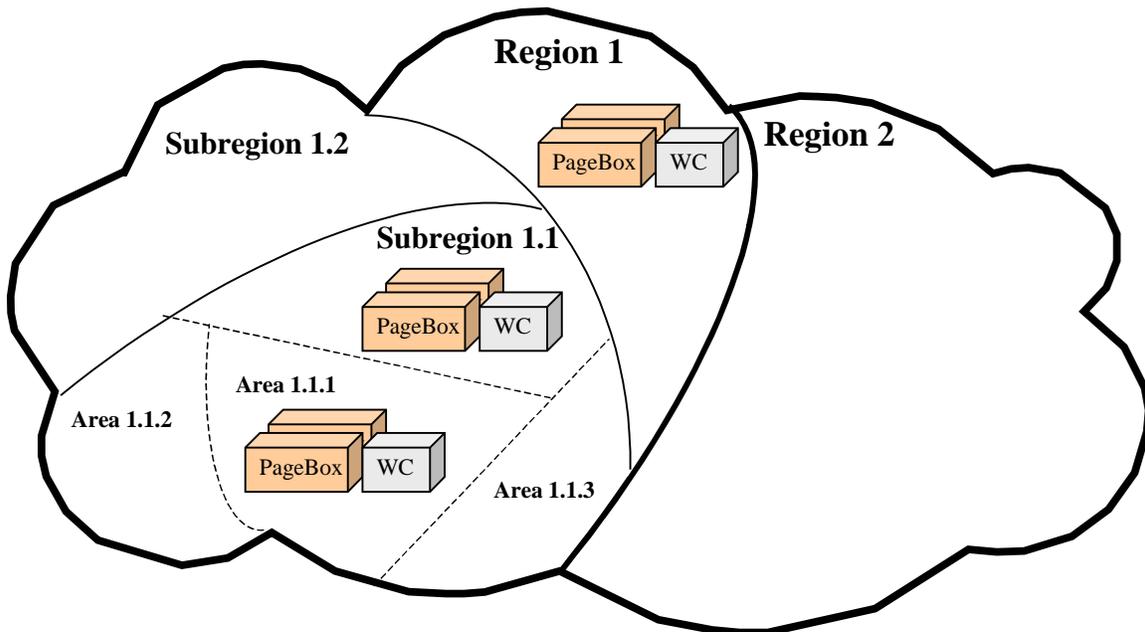


Figure 6: Areas

Suppose archive 1 has a high quality of service requirement or is heavily used. The ISP deploys it on areas.

If it has a low quality of service requirement, the ISP deploys it on regions. In intermediate cases, the ISP deploys it on sub-regions.

### **3.3.2.2 Session handling**

#### **3.3.2.2.1 Principle**

PageBox doesn't require deployed Java servers to support sessions across instances.

Therefore Web Caches must:

?? Be notified of sessions creation and deletion

?? Once a client has entered a session, always route its messages to the same PageBox

When a session is created, the PageBox multicasts an ICP message containing the Session identifier to Web Caches in the same area. Web Caches add the session identifier and the PageBox IP address to a session table and acknowledge the request.

When a session is invalidated or timed out, the PageBox multicasts an ICP message containing the Session identifier to Web Caches in the same area. Web Caches remove the session from the session table and acknowledge the request.

Session tracking uses either cookies or URL rewriting. If the session table is not empty, a Web cache:

- 1) Scans the URL and the message to find the session identifier
- 2) Uses the session identifier to retrieve the IP address of the managing PageBox
- 3) Routes the message to this PageBox

We recommend using URL rewriting as it is faster to process.
--

If a Web Cache restarts after the session failure, it multicasts messages to PageBoxes in the same area. Then the PageBoxes send the identifiers of their current sessions.

#### **3.3.2.2.2 Implementation in PageBox add-on**

The current PageBox can be extended to support session handling inside unmodified Application Servers.

Instead calling target servlets with the `HttpServletRequest` object of the Application Server, it provides an object that extends the `HttpServletRequest` implementation.

When `getSession()` is invoked, instead returning the `HttpSession` object of the Application Server, it provides an object that extends the `HttpSession` implementation.

When a session is created (when `getSession()` is invoked) PageBox intercepts the creation and multicasts an ICP creation message to Web Caches. When a session is invalidated or timed out, PageBox intercepts it and multicasts an ICP deletion to Web Caches.

#### **3.3.2.2.3 Issue**

Servlet specification states the Session identifier is assigned by the servlet container and is unique and implementation dependent. It however lacks a definition of the scope of this uniqueness. The solution above assumes area uniqueness.

The PageBox add-on solution is also neither elegant nor portable (implementation classes depend on the Application Server). Pluggable Session Management as provided in iPlanet would address both issues.

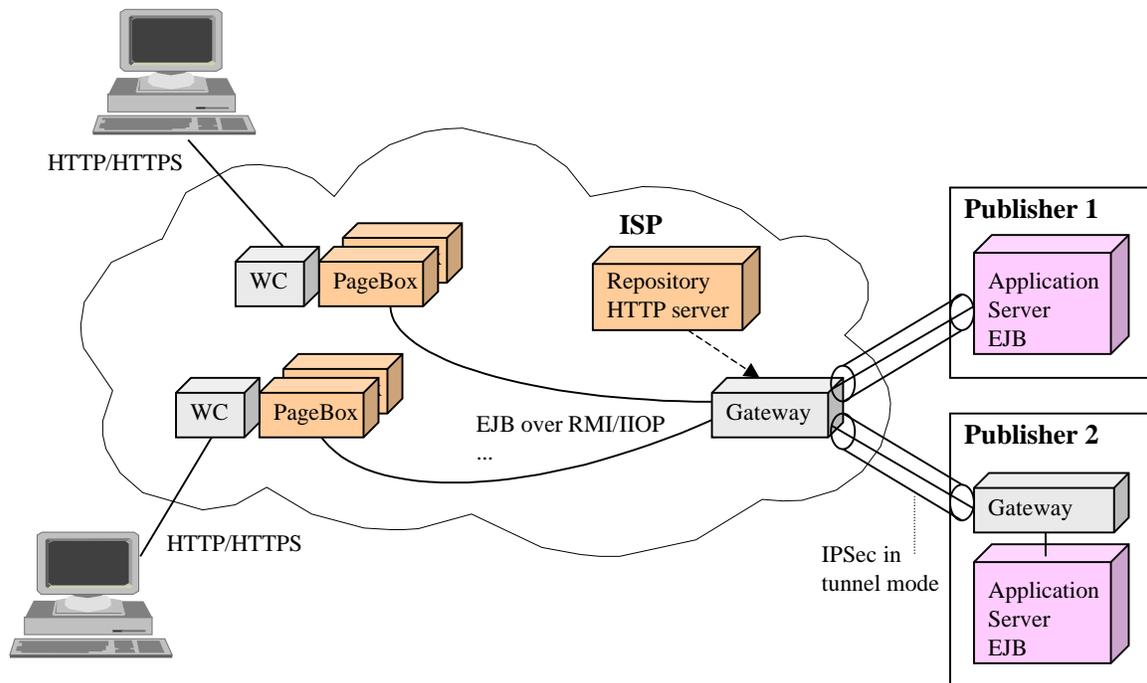
### 3.3.3 Protocols and security

#### 3.3.3.1 *Client/server protocol*

To support client/server requests issued by Published Web Archives,

- 1) The publisher must accept non-HTTP traffic coming from its ISP and carried over IPSec
- 2) The ISP must send this traffic using IPSec

The PageBoxes and ISP internal network is considered as secure.



**Figure 7: Client/server security**

Publishers are connected to Entry Points gateways that establish an IPSec tunnel with Publisher hosts or IPSec gateways. The creation of this tunnel implies to establish a security association (SA), so involves the use of Internet Key Exchange (IKE) between the gateway and the publisher.

The IKE authentication is performed using RSA public key encryption.

The repository HTTP entry point described in the coming 3.3.3.2 Archive publication section can automatically configure the gateway.

#### 3.3.3.2 *End user security*

The ISP can provide an HTTP over SSL access to End-Users.

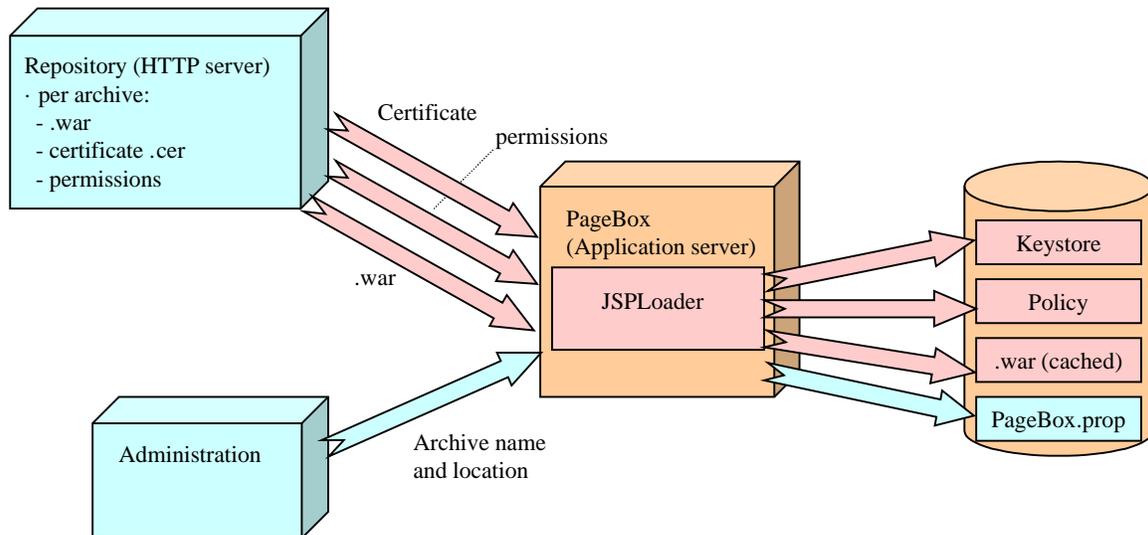
In this case, the publisher must specify:

1. If it requires SSL.  
If it needs to authenticate the server, it cannot assume the ISP uses the same server certificate chain in different PageBoxes. However all PageBoxes certificate chains must include the certificate of a CA the publisher trusts
2. If it need to authenticate clients using certificates.  
Then the publisher is responsible to implement client certificates checking in its archive.

### 3.3.4 Archive publication and distribution

#### 3.3.4.1 *Archive distribution*

The existing PageBox mechanism is reused.  
It separates control and data flow:



**Figure 8: PageBox distribution**

The implementation uses a simple HTTP GET request (command flow) to send the archive name and location. It can be replaced by an ICP request multicasted to PageBoxes.

PageBox stores the archive location in a PageBox property. JSPLoader uses the archive location to download (data flow)

- ?? The archive certificate
- ?? Permissions of the sandbox where the archive will run
- ?? The archive itself (.war)

JSPLoader stores the certificate in a key store, adds keystore and grant lines to the Permissions to generate a well-formed policy file, uses this policy and key store to create classes in the sandbox. It also stores a local copy of the archive.

It checks if the certification chain includes a trusted CA certificate and if the certificate is not revoked using JNDI / LDAP.

For convenience these files are stored in the same directory on the repository and named:

- ?? *Archive.war* archive
- ?? *Archive.cer* certificate in base 64
- ?? *Archive.policy* permission list

This mechanism is convenient for distribution. The ISP

- ?? Maintains a set of repository HTTP or FTP servers to serve the archives, permissions and certificates. Requests can be load balanced between servers using DNS round robin
- ?? Commands the archive installation or update of the archives

Archive installation and update being synchronous, the process can be completely automated and run by batches. It is fully dynamic. PageBoxes can also be remotely monitored. The data

being locally copied, the unavailability of a repository server can at most delay updates and there is no risk download burst to collapse the network.

The administration should maintain a base containing the list of the archives installed per PageBox and containing for each archive:

- ?? Its status: to install, deployed or in quarantine
- ?? The repository server it should be downloaded from
- ?? Its installation date
- ?? Its scheduled deployment date
- ?? Its scheduled expiration date

For simplicity, it can be a simple XML file, replicated on different servers:

```
<archive>
  <name>myArchive</name>
  <downloadFrom>http://myServer/myDir</downloadFrom>
  <installDate>Wed, 01 Nov 2000 11:24:15 GMT</installDate >
  <status>deployed</status>
  <deployDate>Wed, 29 Nov 2000 11:24:15 GMT</deployDate >
  <expireDate>Thu, 29 Nov 2001 11:24:15 GMT</expireDate >
</archive>
...
```

It allows automatic deployment and deletion.

The deletion is implemented like distribution using an HTTP GET or ICP request with the archive name. PageBox removes the local archive copy and unreferences archives objects, classes and class loader. The process is fully dynamic.

#### **3.3.4.2 Archive publication**

The publisher must:

- ?? Must be identified
- ?? Publish through a secure channel
- ?? Provide
  - ?? The list of the permissions its archive requires. The ISP can offer only a predefined list of permission sets
  - ?? The certificates its archive uses.
    - It must use archive certificate chains that include a CA the ISP trusts
  - ?? The quality of service requested
  - ?? The expiration date of its publication.
    - The publisher can later postpone its expiration date.
  - ?? IPsec publisher gateway (if any) and application server address, Internet Security Association and Key Management Protocol (ISAKMP) information allowing the archive to establish a communication with the application server over IPsec
  - ?? End Users security requirement (SSL, client authentication)
  - ?? The archive functional requirements (sessions)

The ISP provides an HTTP entry point on its repository. It uses SSL to provide a secure connection and identify the publisher. At SSL handshaking, it issues a Certificate request and the publisher must return its certificate:

```
1. Client hello          ---->
                        <---- 2. Server hello
                        <---- 3. Certificate
                        <---- 4. Certificate request
```

	<-----	5. Server key exchange (Optional)
	<-----	6. Server hello done
<b>7. Certificate</b>	----->	
8. Client key exchange	----->	
9. Certificate verify (Optional)	----->	
10. Change cipher spec	----->	
11. Finished	----->	
	<-----	12. Change cipher spec
	<-----	13. Finished
14. Encrypted data	<-----	14. Encrypted data

Then the publisher uses the SSL connection to send the archive name, information listed above and the archive itself.

The ISP can act as a Certificate Authority or trust a Certificate Authority. It can support different client certificate qualities.

A publication can fail:

- ?? because the publisher asks for permissions not granted for its client certificate quality
- ?? because the archive is too big for this client certificate quality

#### **3.3.4.3 Charging model**

Based on the client certificate and its non-repudiation ability, the ISP can charge the customer depending on:

- ?? The archive size
- ?? The requested permissions
- ?? A requested quality of service – based on PageBox host resources and on the number of PageBox where the archive is deployed
- ?? End Users security requirement (SSL, client authentication)
- ?? The archive functional requirements (sessions)
- ?? The duration of the publication

The ISP can provide a fully automated service where the customer:

- ?? Is prompted for its credit card number and eMail address
- ?? Is granted a certificate
- ?? Is prompted for the expiration date of its publication.
- ?? Is prompted for the archive location and permissions
- ?? Is prompted for the IPSec publisher gateway (if any) and host address, Internet Security Association and Key Management Protocol (ISAKMP) information allowing the archive to establish a communication with the application server over IPSec
- ?? Can be prompted for End Users security requirement (SSL, client authentication)
- ?? Can be prompted for Session support

The ISP can provide a service similar to ejip.net console (<http://www.ejip.net>).

#### **3.3.4.4 Legal aspects**

The publisher must include in its archive all libraries needed for the communication between the Web Archive (presentation) and the Application Server but if the ISP choose to deploy these libraries on its PageBoxes.

If the publisher includes commercial libraries, it is responsible to get the needed licenses.

The ISP can check the behavior of an archive on quarantine PageBox(es) before full deployment. It is also true for updates. The ISP must notify the publisher of the update acceptance (end of the

quarantine). It is the responsibility of the publisher to support both updated and non-updated presentations during the quarantine.

### **3.3.4.5 *Archive transformation***

The publisher publishes a war archive with non-compiled JSP.  
The ISP distributes a war archive with compiled JSP to PageBoxes.

This mechanism:

1. Allows the publisher and the ISP to use different Java Application servers
2. Allows the ISP to identify compilation errors on a central point
3. Allows PageBoxes to deliver a more consistent response time
4. Reduces PageBox memory requirement

---

### **3.3.5 *Reference data***

PageBox enables the deployment and the use of reference data on ISP site. Their use is recommended.

Reference data are data:

- ?? Accessed in read only
- ?? Unlikely to change on long period of time

They can be used:

- ?? To present data in a human readable form (as the human readable form doesn't cross the network, traffic is reduced)
- ?? For meta data (for instance Presentation customization) and component serialized state
- ?? To address internationalization needs
- ?? To assist the user
- ?? To perform validity checking

They

1. Enhance user productivity
2. Minimize the round trip number to the publisher site, reducing the bandwidth requirement and improving the response time

PageBox supports different reference data mechanisms.

#### **3.3.5.1 *Serialized objects***

PageBox class loader retrieve resources:

1. From the archive
2. If the resource is not in the archive, from the archive download location

Suppose a developer can represent its reference data with an HashMap. She or he can populate it with:

```
InputStream is = getClass().getClassLoader().getResourceAsStream("myResource.ser");
ObjectInputStream ois = new ObjectInputStream(is);
HashMap hm = (HashMap) ois.readObject();
is.close();
```

#### **3.3.5.2 *JMS***

The publisher includes a JMS client library in its archive or relies on an ISP provided library.

Steps:

1. The archive code subscribes to a publisher-hosted topic

2. When the publisher wants to update the reference data it publishes a message to the topic containing the new reference data
3. The archive is notified and can update its reference data from the message

---

### 3.3.6 Local data update

The ISP can forbid local disk access, offer a disk space or database quota.

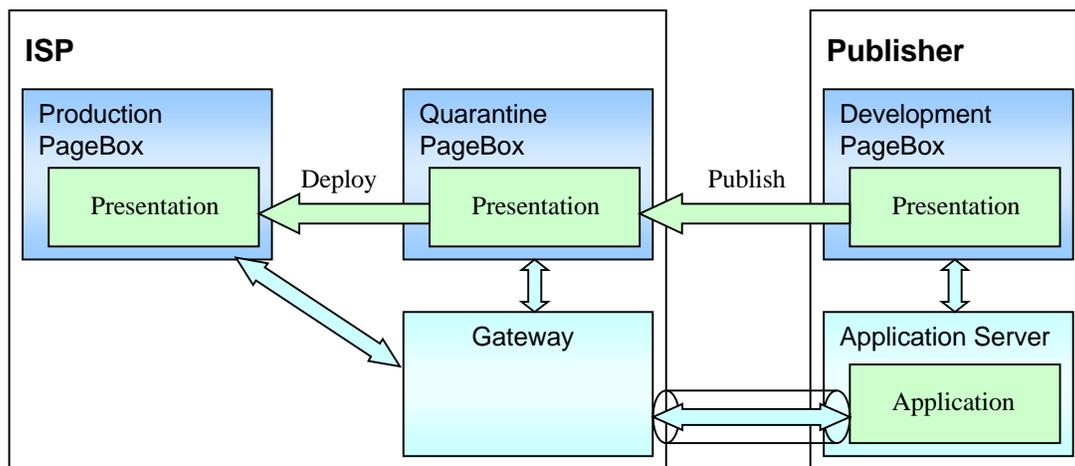
Local data update is not recommended, as the ISP doesn't guarantee a given user will always be served by the same PageBox.

Local data update can however be used to record user requests when the publisher application is unavailable.

For completely dehosted solution the publisher can combine the use the ISP PageBox and of an ASP hosted application server. The later service is already provided by companies such as eijp.net ([www.eijp.net](http://www.eijp.net)) and use similar publication mean.

---

### 3.3.7 Life cycle



**Figure 9: life cycle**

PageBox is today an Application Server add-on and can become an Application Server feature. Programmers can develop in a PageBox. In the future, PageBox support can be integrated in J2EE IDEs. IDEs support servlets and JSPs debugging and can support PageBox packaging.

Therefore the preferred development cycle for PageBox presentations is closely the same as for regular presentation:

- ?? Programmers and page authors develop and perform unit test on their workstation, in a PageBox environment, using a centralized change management tool such as CVS
- ?? QA perform stress and regression tests using test tools

Then the Publisher publishes the presentation. The ISP installs it on quarantine PageBox(es).

The publisher acts as a regular user to check the Presentation is properly deployed and pass QA tests.

The ISP checks for harmful behaviors:

- ?? Loitering
- ?? Loops
- ?? Handling of abnormal conditions such as gateway failure

The ISP is not responsible to implement or perform Presentation tests. However it gets statistics on uncalled classes (coverage), number of invocations, memory and CPU use.

The ISP can accept to deploy the presentation only when it has been extensively tested.

---

### ***3.3.8 Troubleshooting***

To be effective PageBox infrastructure must allow and promote cooperation of ISP and Publisher for troubleshooting through automated means and well defined responsibilities.

The ISP is responsible to fix hardware and network problems, to identify and fix network bottlenecks and resource shortage.

It is also responsible to identify misbehaving presentations (memory leaks, resource overuse) and to report these problems to the publisher. It is allowed to undeploy presentations when the problem has an impact on other presentations.

The publisher must handle problems raised by the ISP and provide fixes or circumvention. It can publish an archive update referring to a problem number and the ISP can choose to deploy it either with a shortened quarantine period or no quarantine at all.

Users report problems to publisher that can be:

?? Response time or availability problems

?? Functional problems

The publisher is responsible to identify and in some cases to report the problem to the ISP. The ISP must provide it access to statistics and logs of the failing PageBox.

The main communication mean between the ISP and the publisher and between the end user and the publisher is SMTP mail. It allows automatic generation and parsing and provides secured delivery. For instance, ISP robot can control PageBoxes, identify the misbehaving presentation and submit a mail to the presentation publisher.

PageBox provides servlets to display and administrate statistics and logs.

There are using GET mode to simplify invocation from batches and scheduling tools.

Both statistics and logs are persistent and survive crashes.

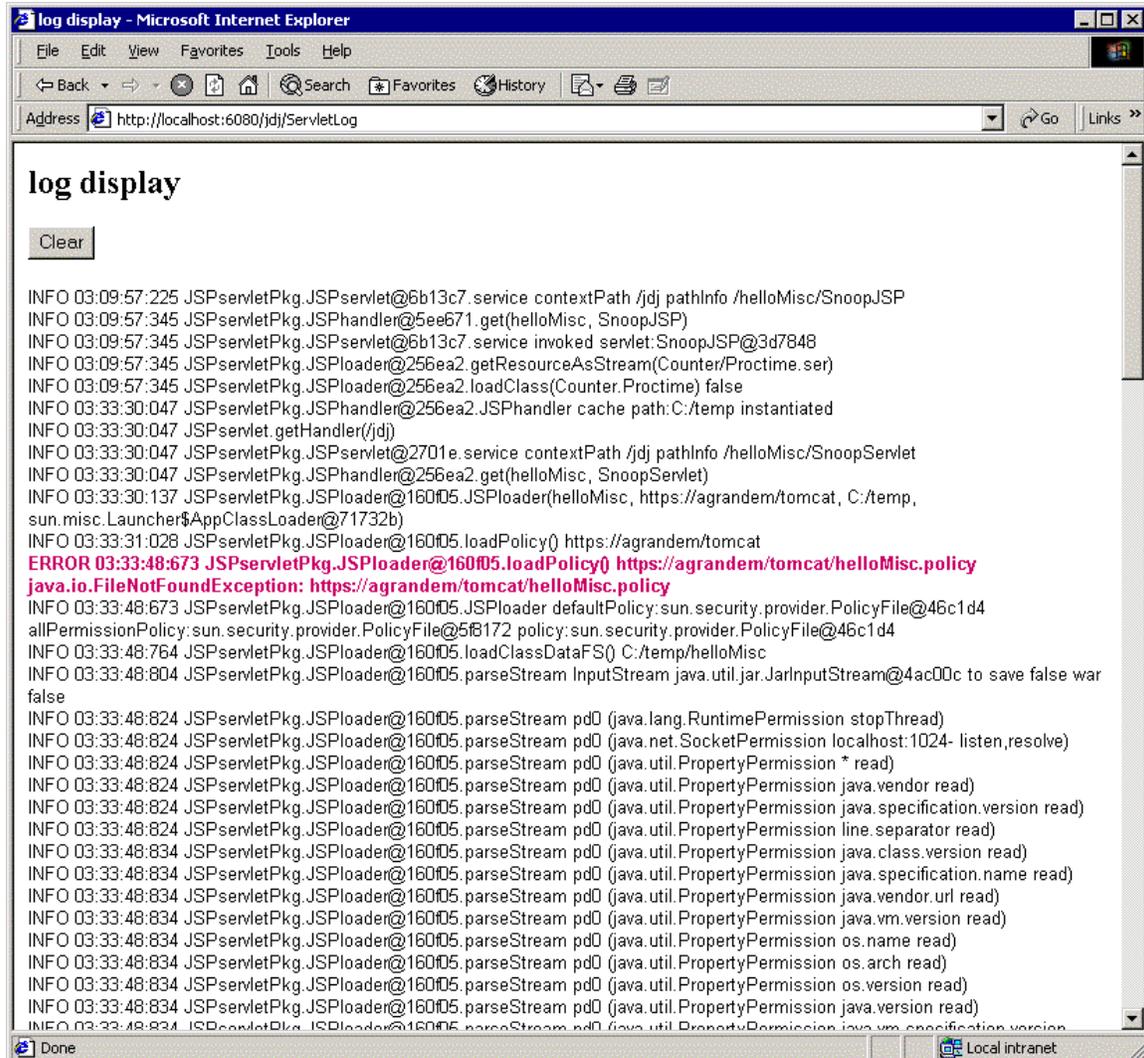
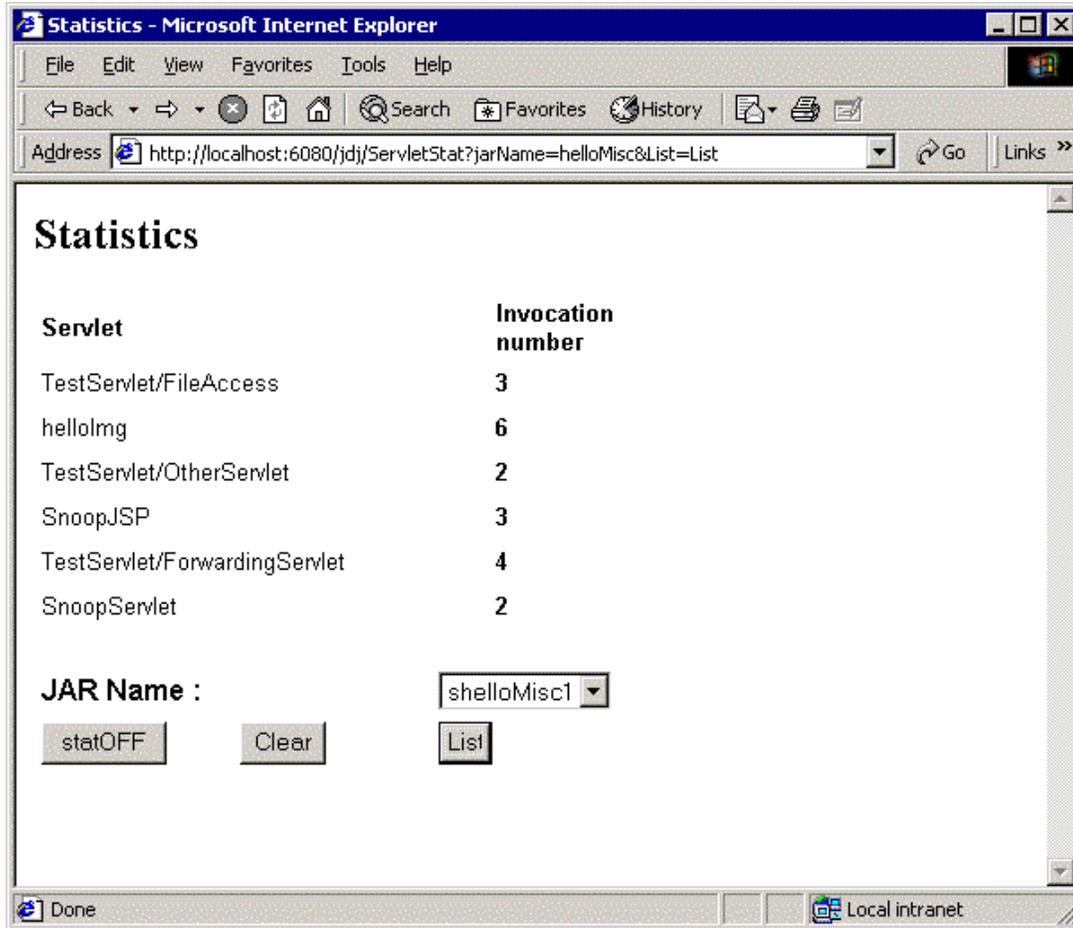


Figure 10: PageBox log display



**Figure 11: PageBox Statistics**

Statistics can also be returned in a comma separated (.csv) format more suitable for post processing.

The ISP uses these servlets to remotely administrate PageBoxes. When it identifies a Presentation problem on a PageBox, it gives a temporary access to the failing PageBox administration servlets to the publisher and emails it the URL to use.

The publisher can also ask for a temporary access to troubleshoot functional problems raised by their users using email.

Troubleshooting is the only reason PageBox needs an API we present in the next section.

### **3.3.9 PageBox API**

The PageBox API has two purposes:

1. Giving access to a String that uniquely identifies the PageBox.  
This field has a meaning only for the ISP, which is responsible to set it in PageBox web.xml. The publisher is responsible to define a mean to display this identifier to the user. The publisher uses it to ask for a PageBox access.
2. Offering a logging facility the publisher can use for diagnosis

The PageBoxAPI class provides therefore two methods:

1. `getID()` that returns the unique identifier of a PageBox

2. `userprint(String message)` that logs a message.  
`userprint` adds a timestamp and the servlet or JSP path to the message.

Two ISPs cannot use the same identifier as a PageBox can be deployed on both.

### 3.4 Advantage analysis

#### 3.4.1 *Traffic*

In this section, I evaluate the benefit of PageBox in term of traffic, using a simple example of a large dynamically generated page. The PageBox advantage could be confirmed by other studies.

I found on this page the following results:

?? displayed: 30,230 characters

?? data: 9,398 characters

?? page size: 233,305 characters

The data represents 4% of the transferred message.

This page is mainly made of 170 lines containing 15 fields that could be described in xml by:

```
<flightlist>
  <flight>
    <price>xxxx</price>
    <to>
      <airline>xxx</airline>
      <stopnb>x</stopnb>
      <airport>xxx</airport>
      <day>xx</day>
      <month>xx</month>
      <year>xx</year>
      <departure>
        <hour>xx</hour>
        <min>xx</min>
      </departure>
      <arrival>
        <hour>xx</hour>
        <min>xx</min>
      </arrival>
      <duration>
        <hour>xx</hour>
        <min>xx</min>
      </duration >
    </to>
    <return>
      <airline>xxx</airline>
      <airport>xxx</airport>
      <day>xx</day>
      <month>xx</month>
      <year>xx</year>
      <departure>
        <hour>xx</hour>
        <min>xx</min>
      </departure>
      <arrival>
        <hour>xx</hour>
```

```

<min>xx</min>
</arrival>
<duration>
  <hour>xx</hour>
  <min>xx</min>
</duration >
</return>
</flight>
<flightlist>

```

It represents 478 characters per line<sup>1</sup>, so an overhead of 81,260 characters.

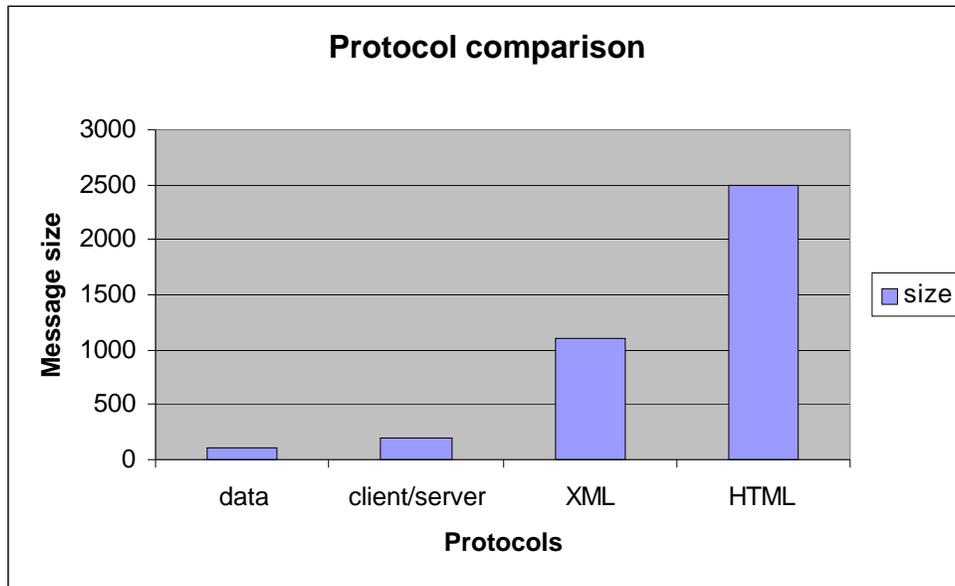
In this case we also need:

1. To include non-XML (HTML summary) data and the XSL transformation invocation in the page
2. To download XSL data

An XML solution can use half less bandwidth than HTML<sup>2</sup>.

The Java serialization of these 170 lines requires 19,138 bytes<sup>3</sup>, so an overhead of 9,740 bytes.

We can summarize these results on the chart below.



**Figure 12: Protocol comparison**

Data size being 100, client/server requires 200, XML 1100 and HTML 2500. Using client/server protocol requires twelve times less bandwidth than HTML and five times less than XML.

PageBoxes use HTML with the consumer equipment and client/server protocol with the publisher. As a result,

1. The publisher needs to subscribe for an ISP link ten times smaller than with HTTP

<sup>1</sup> At least, without blanks, tabs and new lines. To remain readable XML message is even bigger – 560 bytes with Windows CR/LF in that case.

<sup>2</sup> Assuming XSLT and XML parsing means are already installed on the browser.

<sup>3</sup> I coded the XML representation and the Java class as a mean developer. In the Java class, dates are represented by longs and line sequence by a Vector.

2. The ISPs also need smaller infrastructure

---

### **3.4.2 DSL and Cable network**

The advent of fast local loop has significant impacts:

1. It is more a quantum leap than the smooth evolution of former technologies
2. The number of connected devices is also likely to increase. Will the infrastructure be able to evolve at the same rate as connected equipment or will we face the same situation for network as for hardware where workstation performances are of the same level of magnitude as expensive servers?
3. Consumer devices are no longer oversized for their networking capabilities. It is especially true for older devices

PageBox is an enabling technology, allowing:

1. To meet Consumer increasing expectations
2. To avoid infrastructure and publisher bottlenecks

Though PageBox is primarily designed to address business need for sub-second response time, it can also address similar end consumer needs.

---

### **3.4.3 Markets**

We already listed:

1. Corporate Intranets: the company deploys and operates PageBoxes
2. Companies with large agency/office network: the company relies on ISP for PageBoxes. The agencies can be owned by the company (banks) or not (Travel Agents connect to Global Distribution Systems but are not owned by them)
3. End consumers for interactive applications requiring sub-second response time. They also use ISP services

In these cases, the user uses her or his browser only for HTML or XML presentation.

PageBox can handle WML for WAP devices. Mobile network support is the same as regular ISP support. It can be an important market because:

1. Mobile devices lacks resources and can only run simple interpretation and scripting
2. The services they access must be fast to be effective

PageBox can also address kiosk needs. In this case a set of kiosks is installed in a location such as an airport or a railway station. If the number of kiosks is large enough the kiosk operator can deploy and operate its own PageBoxes as in case 1, otherwise it uses ISP hosted PageBoxes. Downloaded pages call applets to handle kiosk devices such as magnetic and smart card readers.

With PageBox, Presentation and Application are physically separated. It promotes a specialization of companies either as Application providers or Presentation providers. Presentation providers can develop Presentations to legacy IP-enabled applications or to multiple applications.

## 4 Standard need

### 4.1 PageBox

Today PageBox is implemented as a regular Web Archive.  
 It implies it replicates some functions of Application Servers  
 ?? class loader  
 ?? static resource handling

It should replicate web.xml parsing (I confess it doesn't handle it today).

Though this public domain implementation has a value in an interim phase, vendors could better support Application Servers implementing its functions, mainly:

?? Archive download  
 ?? Sandboxes

The PageBox hardware and software requirements are relatively low:

?? An Operating System where Java 2 is available  
 ?? A Java Runtime Environment  
 ?? A Java server supporting servlets and JSPs  
 ?? Though I designed PageBox to favor response time, I found it could handle 9000 simple JSP with 60 MBytes

So a combination of hardware and software could be packaged in a PageBox appliance.

We need to describe PageBox features in Java™ 2 Platform Enterprise Edition Specification.

### 4.2 ICP

RFC 2186 and 2187.

In PageBox integration section, we saw existing ICP\_OP\_QUERY, ICP\_OP\_HIT and ICP\_OP\_MISS were addressing the primary need.

The definition of new messages or the modification of existing messages would however allow PageBoxes to return:

?? A load factor, the Web Cache could use for load balancing  
 ?? A bulk transfer of supported URLs. The message would remain small and the Web Cache would have much less ICP messages to send

We need new ICP messages to handle sessions:

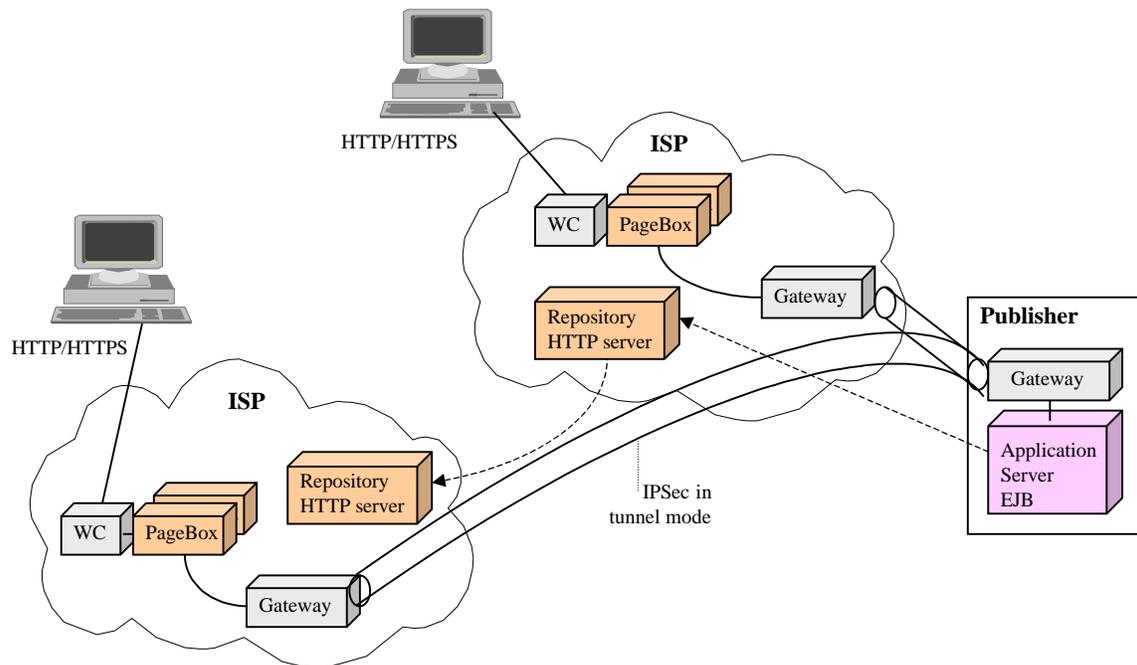
?? ICP\_OP\_SESSION\_ADD issued by PageBoxes. Its payload contains one or many Session identifiers separated by commas.  
 ?? ICP\_OP\_SESSION\_ADD\_ACK issued by Web Caches to acknowledge a ICP\_OP\_SESSION\_ADD  
 ?? ICP\_OP\_SESSION\_DEL issued by PageBoxes. Its payload contains one or many Session identifiers separated by commas.  
 ?? ICP\_OP\_SESSION\_DEL\_ACK issued by Web Caches to acknowledge a ICP\_OP\_SESSION\_DEL

As ICP is used to drive PageBoxes, it can also be used to install, update or delete PageBoxes archives or to get their status:

- ?? ICP\_OP\_ARCHIVE\_UPDATE issued by distribution tool. Its payload is made of lines containing archive name and location separated by commas. When a PageBox receives this message, it updates its property file and downloads the archive(s). If the archive was already installed, it is an update. Otherwise it is an installation.
- ?? ICP\_OP\_ARCHIVE\_UPDATE\_ACK issued by PageBoxes to acknowledge a ICP\_OP\_ARCHIVE\_UPDATE
- ?? ICP\_OP\_ARCHIVE\_DEL issued by distribution tool. Its payload is made of lines containing archive name and location separated by commas. When a PageBox receives this message, it updates its property file and remove the archive(s)
- ?? ICP\_OP\_ARCHIVE\_DEL\_ACK issued by PageBoxes to acknowledge a ICP\_OP\_ARCHIVE\_DEL
- ?? ICP\_OP\_ARCHIVE\_QUERY issued by distribution tool
- ?? ICP\_OP\_ARCHIVE\_STATUS issued by PageBoxes to answer an ICP\_OP\_ARCHIVE\_QUERY. It contains the list of installed and loaded archives and of their locations

### 4.3 Publication protocol

The value of PageBox is augmented if a company can publish a Web Archive to a single ISP and get it deployed by many other ISPs.



**Figure 13: Multiple ISP deployment**

For a multiple ISP deployment, we use exactly the same means as for a single ISP. The first ISP, the ISP the publisher has published to and presumably subscribed, acts as a publisher for other ISPs. Let's call it the editor.

When it has received a subscription request, checked the publisher credential, run the Web application in quarantine, it deploys it internally. It also publishes it to other ISPs.

Each other ISP checks its credentials, finds out it is a trusted ISP and deploys the Web Application. Its deployment involves the same steps as for the editor:

?? Archive distribution

?? Gateway configuration to open a tunnel between the ISP gateway and the publisher gateway or host

As a consequence, we need a standard describing the publication protocol.

---

## 4.4 Summary

We need:

?? To add PageBox to Java™ 2 Platform Enterprise Edition Specification

?? An RFC describing the PageBox architecture and the publication protocol

?? An RFC updating the ICP protocol

## 5 Author biography

I used to be the main designer and lead developer of an Intranet solution. At this time I was working for BEA and the customer was a large French bank. It had 2000 agencies and 23000 personal computers and the solution was designed in 1998. So they are differences with Figure 1.

Presentation servers were IIS. They invoked a local Tuxedo, which was used to deliver local services and to invoke central location Tuxedo services. It meant we had to manage and maintain 2000 servers running both IIS and Tuxedo.

I started to work seriously on Application Servers in fall 1999 after moving to Amadeus. During summer 2000, I submitted an article project about class loaders to Java Developer Journal. It accepted the layout. I came back to SG design to illustrate the article and the concept turned to be more exciting than I expected. It is now divided in three parts, the first one illustrating presentation hosting, a second one administration and a third one the security.