Leveraging Solaris Trusted Extensions to Implement Platform Security Services for the Java™ Language

John Weeks

Senior Staff Engineer
Sun Microsystems Federal, Inc.
http://blogs.sun.com/johnw/

TS-1427

Goal of This Talk

Gain some inspiration to develop label-aware Web Services for Solaris™ Trusted Extensions in the Java™ programming language

See how one developer found creative ways to build label-aware web services based on his experimental set of Java code bindings that run on Solaris Trusted Extensions.

You can use these bindings to create your own web services, too!

Agenda

- Multilevel Security Overview
- Demonstration of Label-Aware Web Services
  - Static Labeled HTML Files
  - Labeled XML Tearlines Files
  - Scaled and Labeled JPG Images
- How I Did It: Getting Under the Hood
  - Web Services Prototype Architectures
  - Java Code Bindings for Solaris Trusted Extensions
- Still More Areas to Explore
- Now It’s Your Turn!
- Q&A

Multilevel Security Overview

- Multilevel security is often referred to as MLS
- Uses labels to segregate classified information
  - Lower-level subjects cannot access higher-level objects
  - Higher-level data cannot be written to lower levels
  - Higher-level subjects can “read down” to lower levels
- Implements mandatory access control (MAC)
- Labels relationships determine access control
  - Equal
  - Dominant
  - Strictly Dominant
  - Disjoint

Multilevel Security Overview

- Labels are composed of two parts
  - Classifications (sensitivities) are hierarchical
  - Compartments (categories) are non-hierarchical
- SECRET A B
  - SECRET is the classification
  - A and B are compartments
- Dominance relationships
  - SECRET A B dominates SECRET A and SECRET B
  - TOP SECRET A dominates SECRET A
- Disjoint relationships
  - SECRET A is disjoint from SECRET B

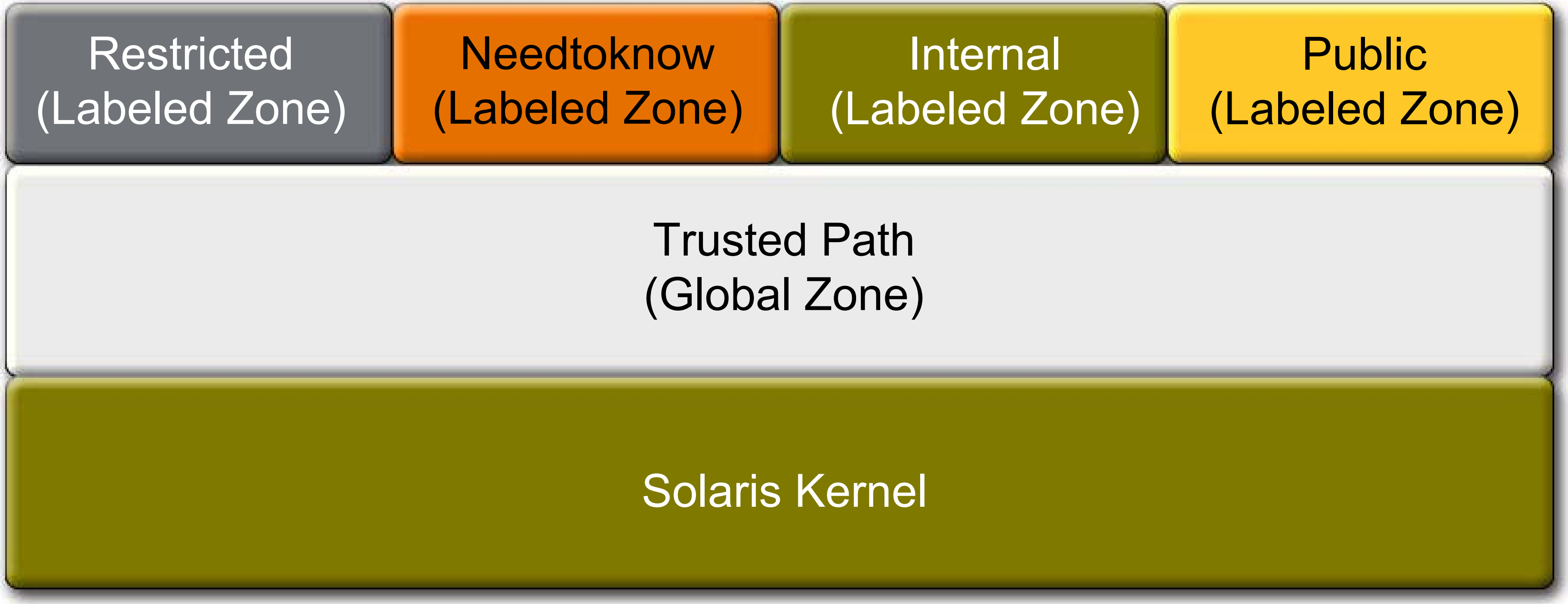
Multilevel Security Overview

- Clearance is upper bound of user's permitted range
  - If user is cleared, the request is approved
  - If user is not cleared, the request is denied
- Label range is a set of labels
  - Bounded by clearance at the upper end
  - Bounded by a minimum label at the lower end
- Ranges are used for sharing multilevel services
  - Multilevel printing
  - Multilevel desktop environment

Multilevel Security Overview

- Multilevel operating systems
  - Solaris Trusted Extensions (Trusted Extensions)
  - Security-Enhanced Linux (SELinux)
- All MLS subjects and objects are labeled
- Trusted Extensions employs Solaris Zones to keep labeled data and processes separate
- Other features of MLS systems include
  - Trusted networking
  - File systems
  - Resource polyinstantiation and resource sharing
  - Multilevel desktop environment

Multilevel Security Overview



Restricted (Labeled Zone)

Needtoknow (Labeled Zone)

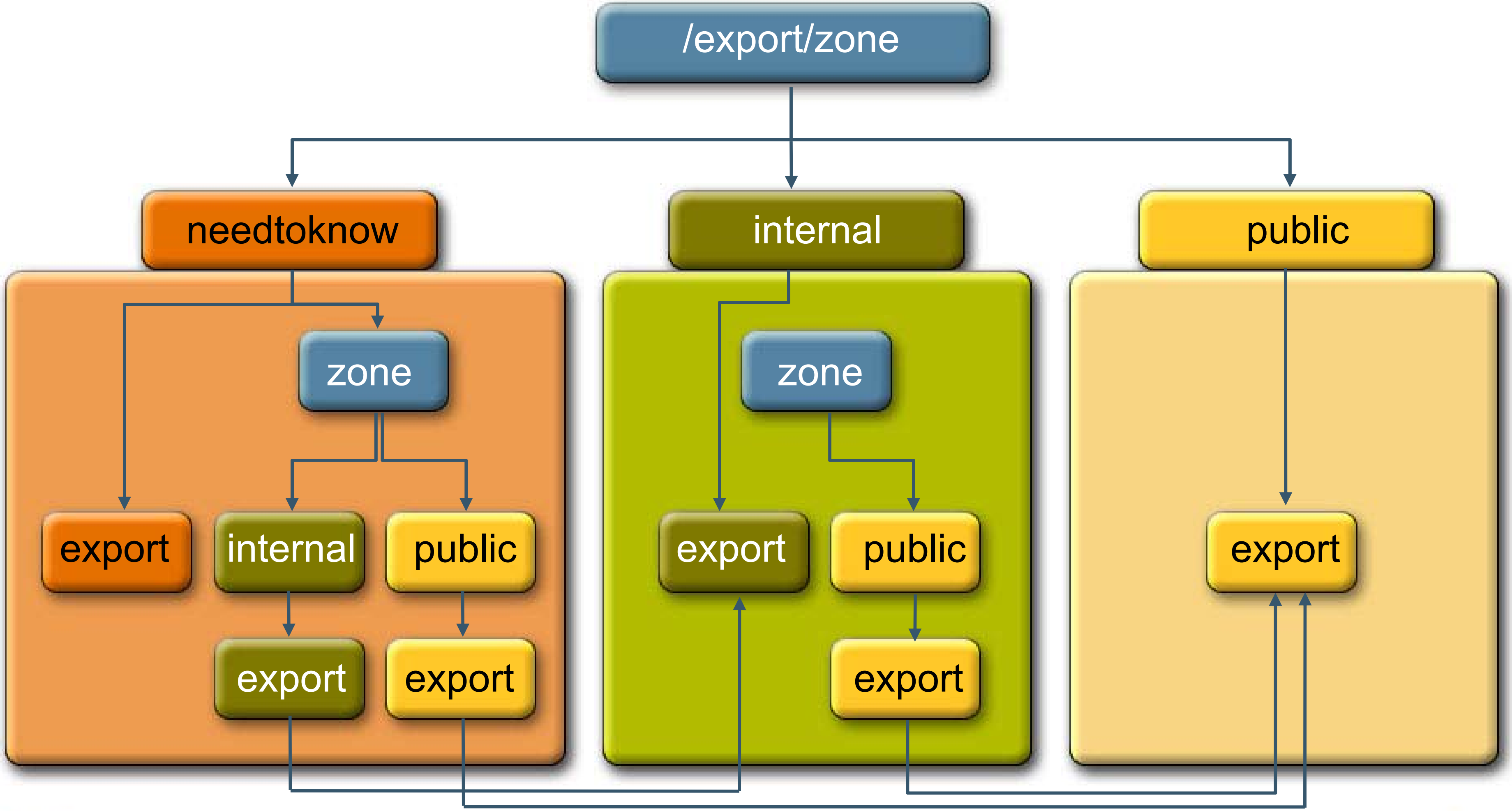
Internal (Labeled Zone)

Public (Labeled Zone)

Trusted Path (Global Zone)

Solaris Kernel

Multilevel Security Overview



DEMO

Serving static HTML and Tearlines labeled XML files and JPG images in a multilevel security environment

Static Labeled HTML File Prototype

- One HTML file per label
- Page shown if the label of the remote connection dominates the label of the HTML file
- 404 error encountered when
  - Request attempts to extrapolate a file name at a higher level
  - Request attempts to access a known file at a higher level

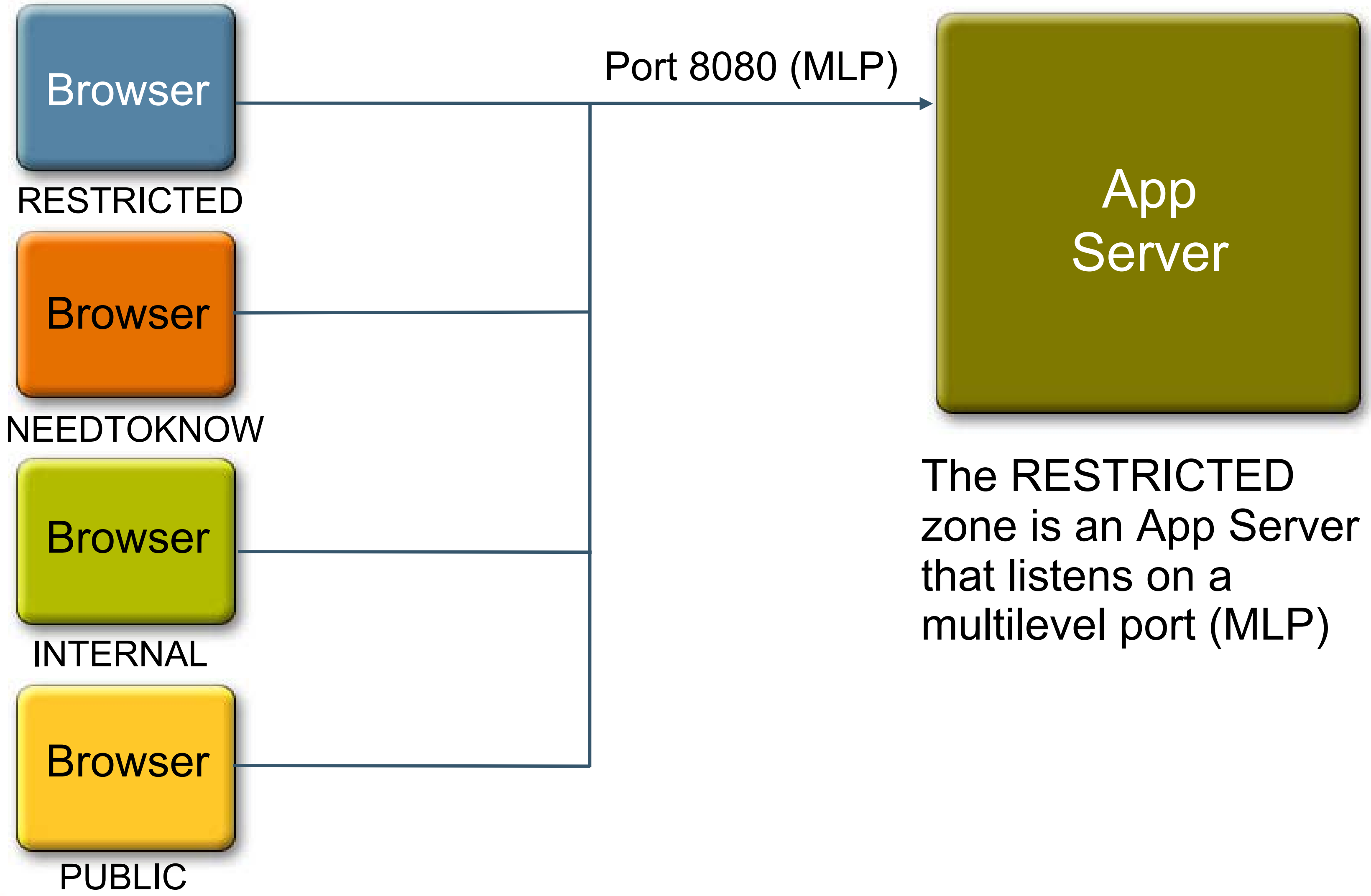
Labeled Tearlines XML File Prototype

- XML file contains data marked at several labels
- Data dominated by connection label is shown
  - XACML requests made
  - Policy used to process requests
  - XACML responses sent
  - Based on “permit” response, filters generate HTML
- Resulting page shows only data the user is permitted to see

Labeled JPG Image File Prototype

- Images shown at different sizes and resolution based on connection label
  - For instance
    - See full image at highest resolution from Restricted connection
    - See smaller image at lower resolution from Internal connection
    - See smaller image at lowest resolution from Public connection
- Image sizing and resolution
  - Dynamically applied to a single image
  - Use Java code interfaces to scale resolution
  - Image stored in highest labeled zone

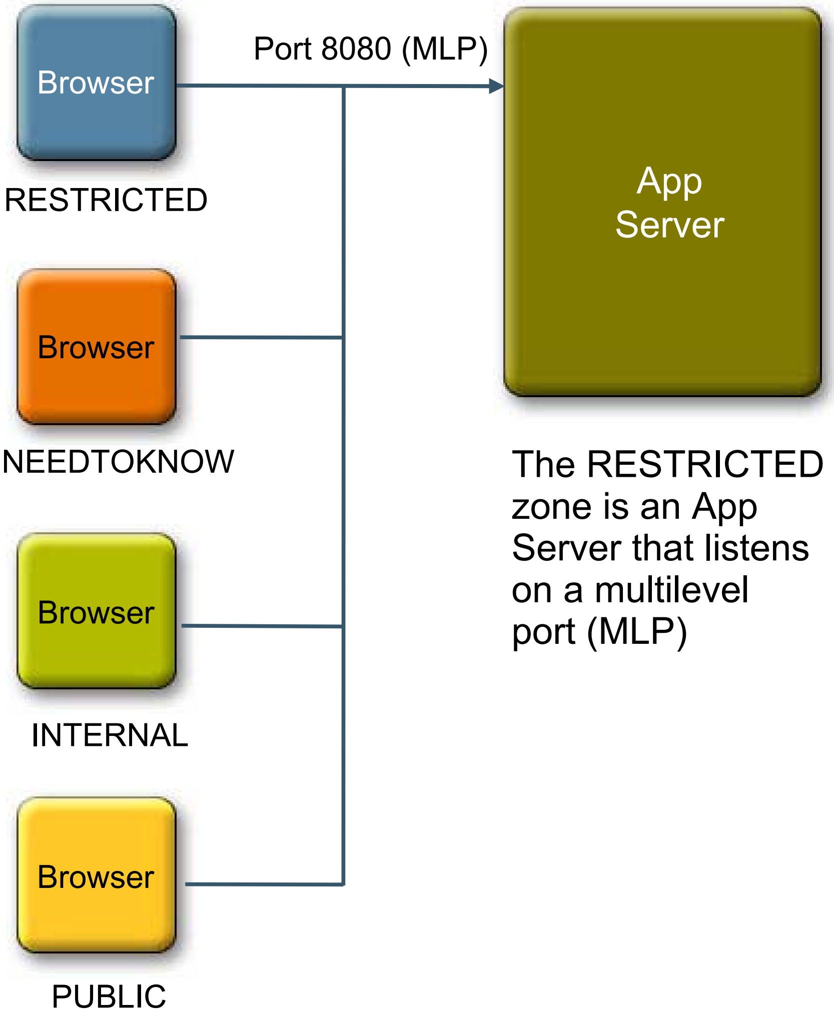
Prototype Architecture—First Try



The RESTRICTED zone is an App Server that listens on a multilevel port (MLP)

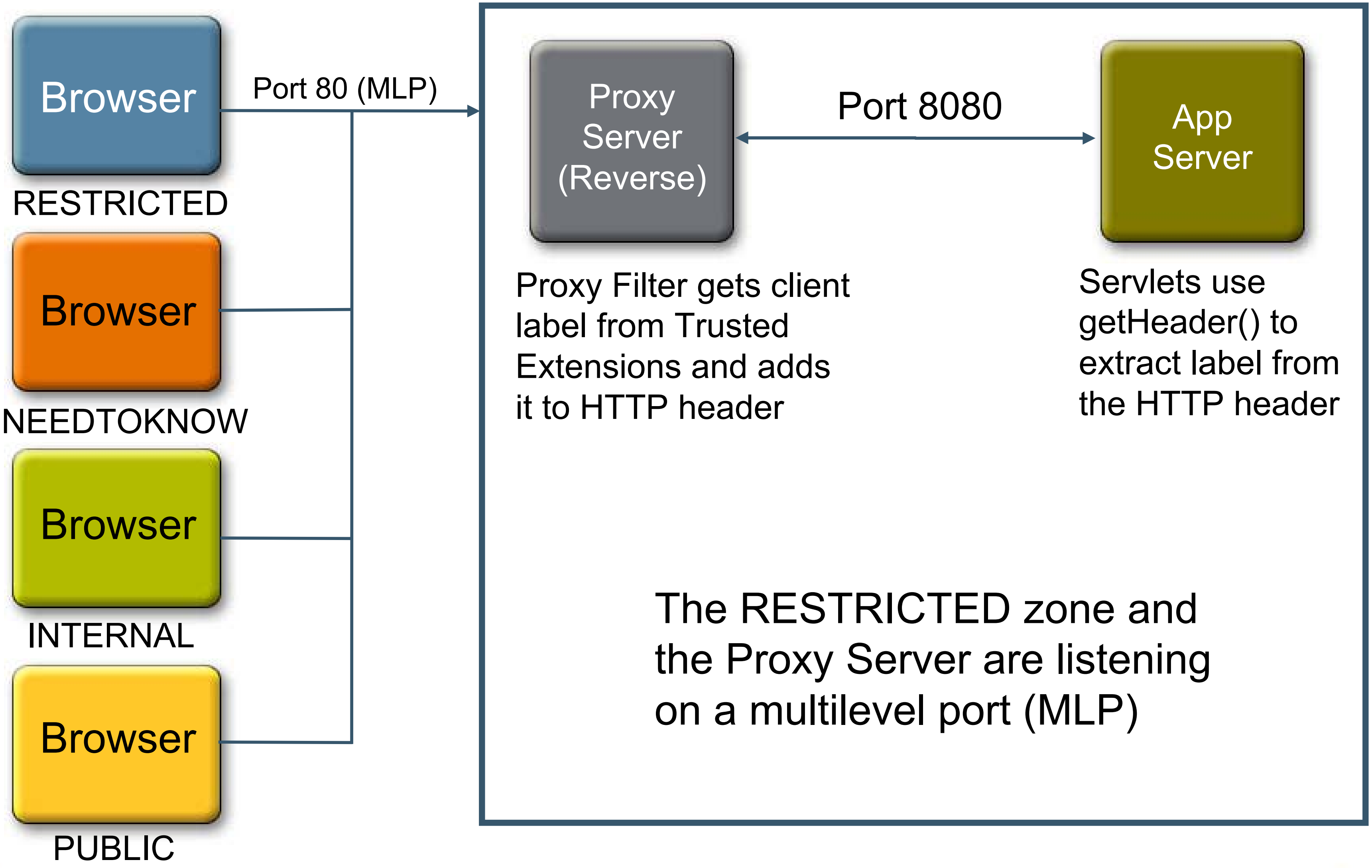
Prototype Architecture—Pros and Cons

- Pros
  - Simple design
  - Enables the app server to listen at multiple labels for browser requests
- Cons
  - Difficult to obtain the peer label for the client connection, which is reached through Socket FileDescriptor



The RESTRICTED zone is an App Server that listens on a multilevel port (MLP)

Prototype Architecture—Second Try



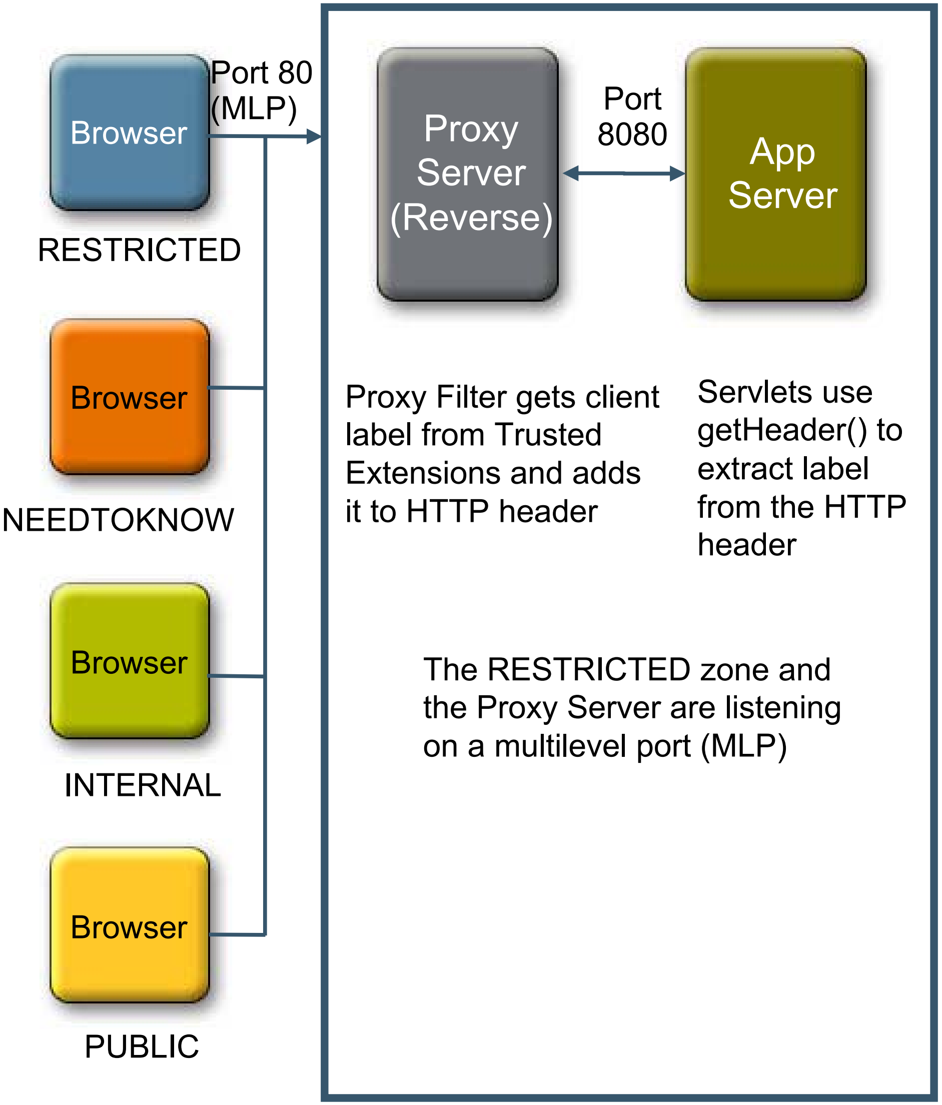
Proxy Filter gets client label from Trusted Extensions and adds it to HTTP header

Servlets use getHeader() to extract label from the HTTP header

The RESTRICTED zone and the Proxy Server are listening on a multilevel port (MLP)

Prototype Architecture—Pros and Cons

- Pros
  - Custom proxy server filter written in C to obtain the peer label
  - Isolates the app server from the actual client connection
- Cons
  - Introduces another component; i.e., proxy server
  - Adds fields to the HTTP header to carry the connection label

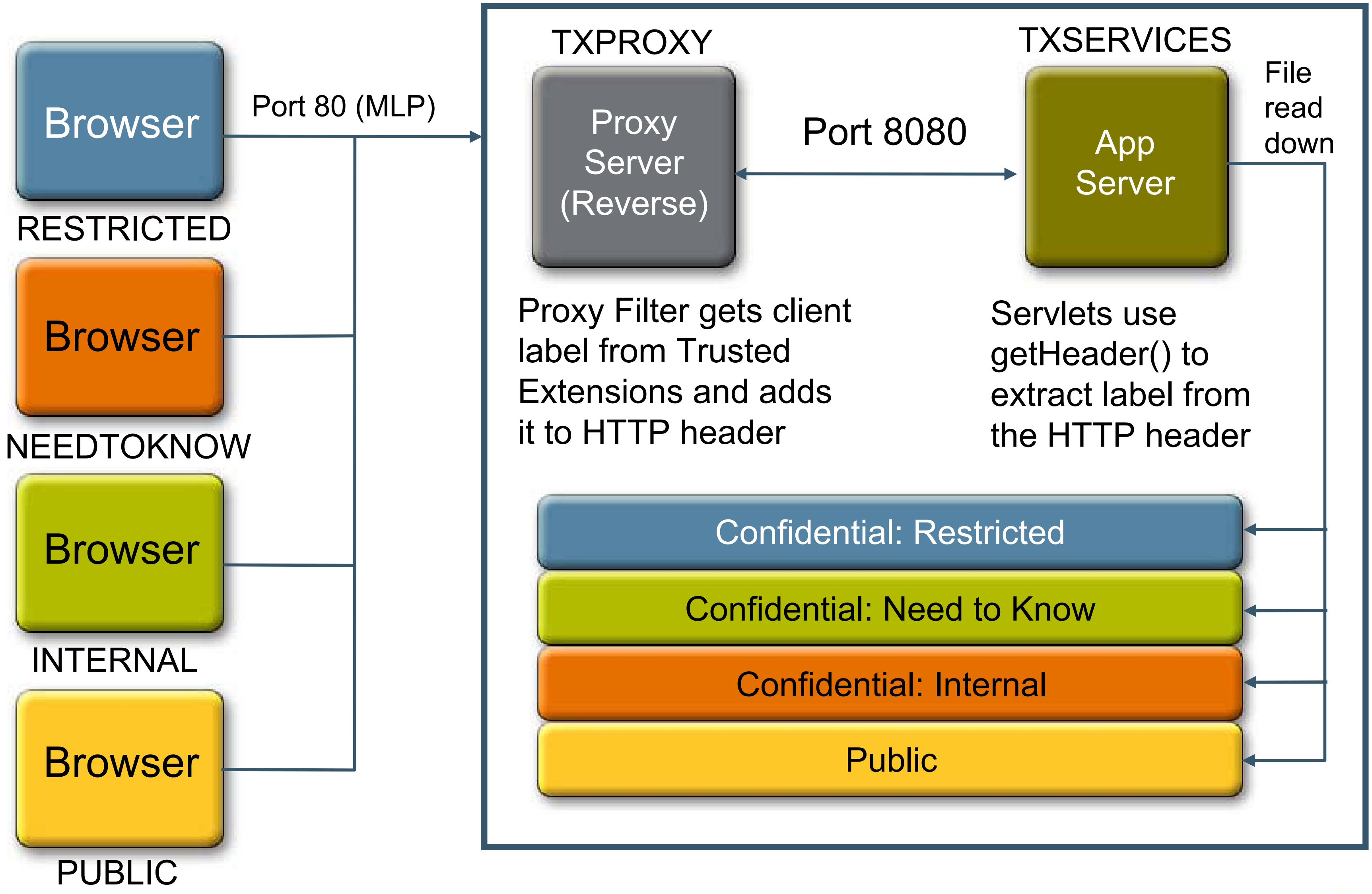


Proxy Filter gets client label from Trusted Extensions and adds it to HTTP header

Servlets use getHeader() to extract label from the HTTP header

The RESTRICTED zone and the Proxy Server are listening on a multilevel port (MLP)

Prototype Architecture—Third Try



Browser

RESTRICTED

Browser

NEEDTOKNOW

Browser

INTERNAL

Browser

PUBLIC

Port 80 (MLP)

TXPROXY

Proxy Server (Reverse)

Port 8080

TXSERVICES

App Server

File read down

Proxy Filter gets client label from Trusted Extensions and adds it to HTTP header

Servlets use getHeader() to extract label from the HTTP header

Confidential: Restricted

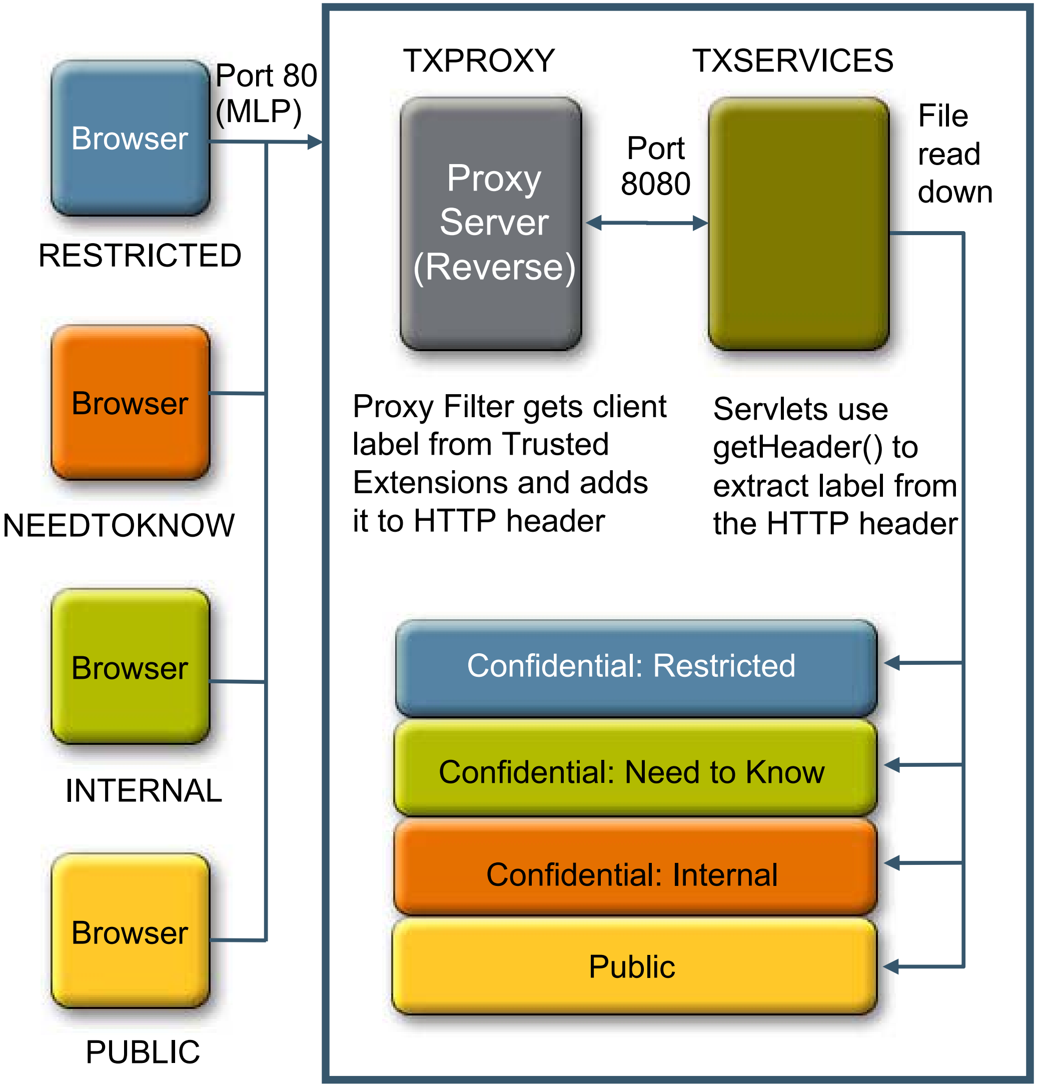
Confidential: Need to Know

Confidential: Internal

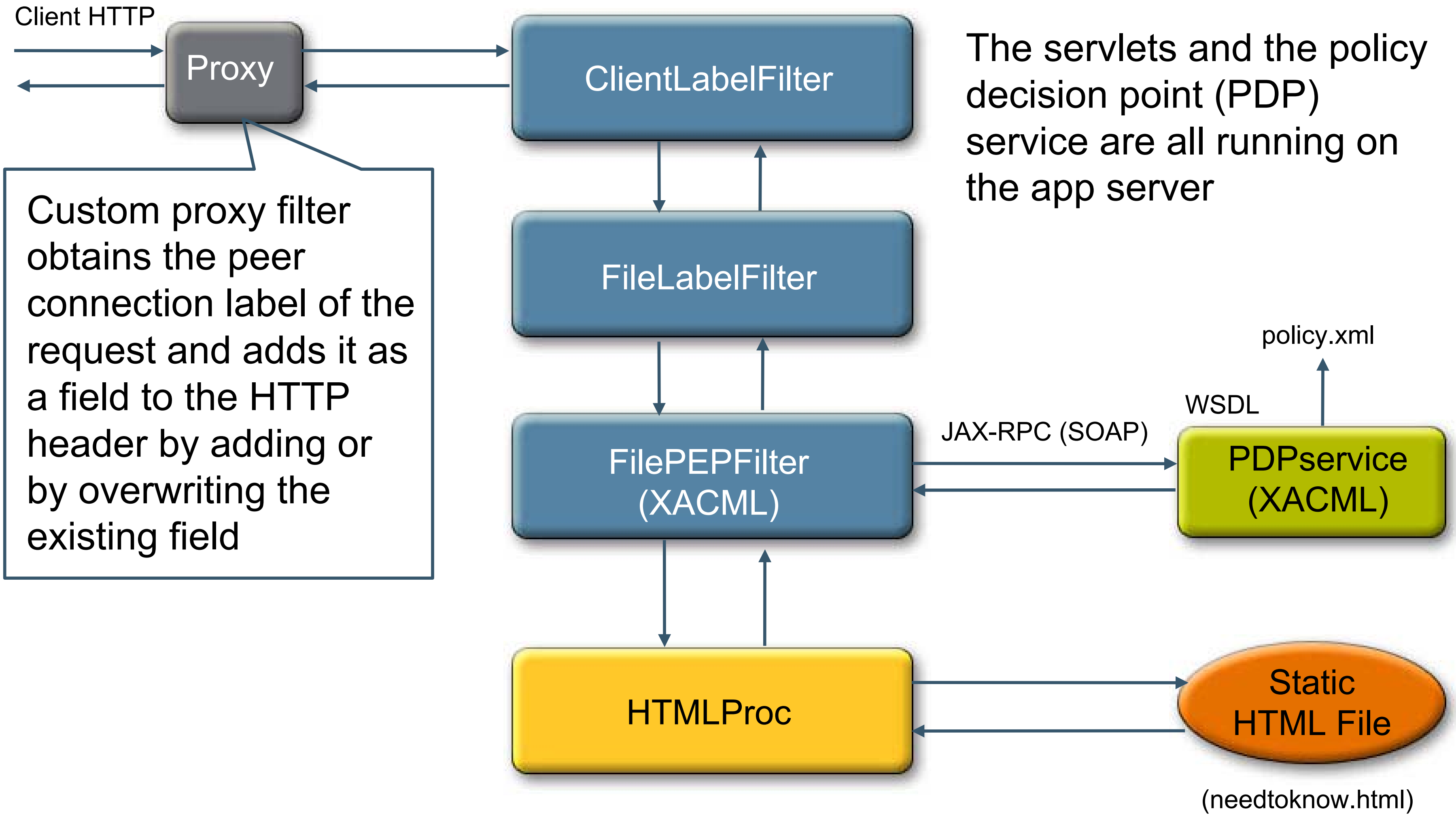
Public

Prototype Architecture—Pros and Cons

- Pros
  - Isolates the proxy server and app server in separate labeled zones (TXPROXY is disjoint from the other labeled zones)
  - App server can only read data
- Cons
  - More complex configuration



Prototype Architecture: HTML

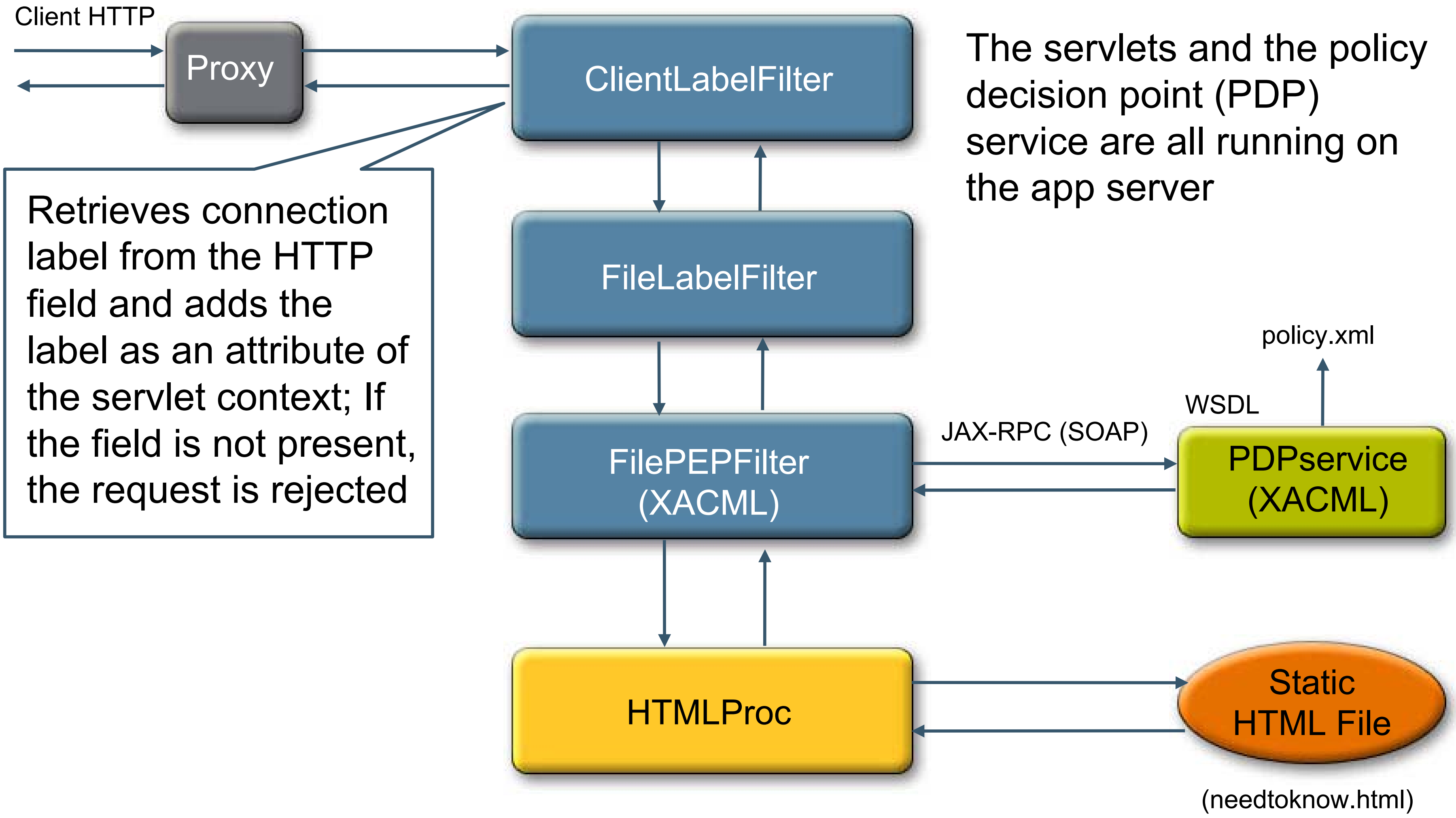


Custom proxy filter obtains the peer connection label of the request and adds it as a field to the HTTP header by adding or by overwriting the existing field

The servlets and the policy decision point (PDP) service are all running on the app server

JAX-RPC = Java API for XML-based RPC

Prototype Architecture: HTML

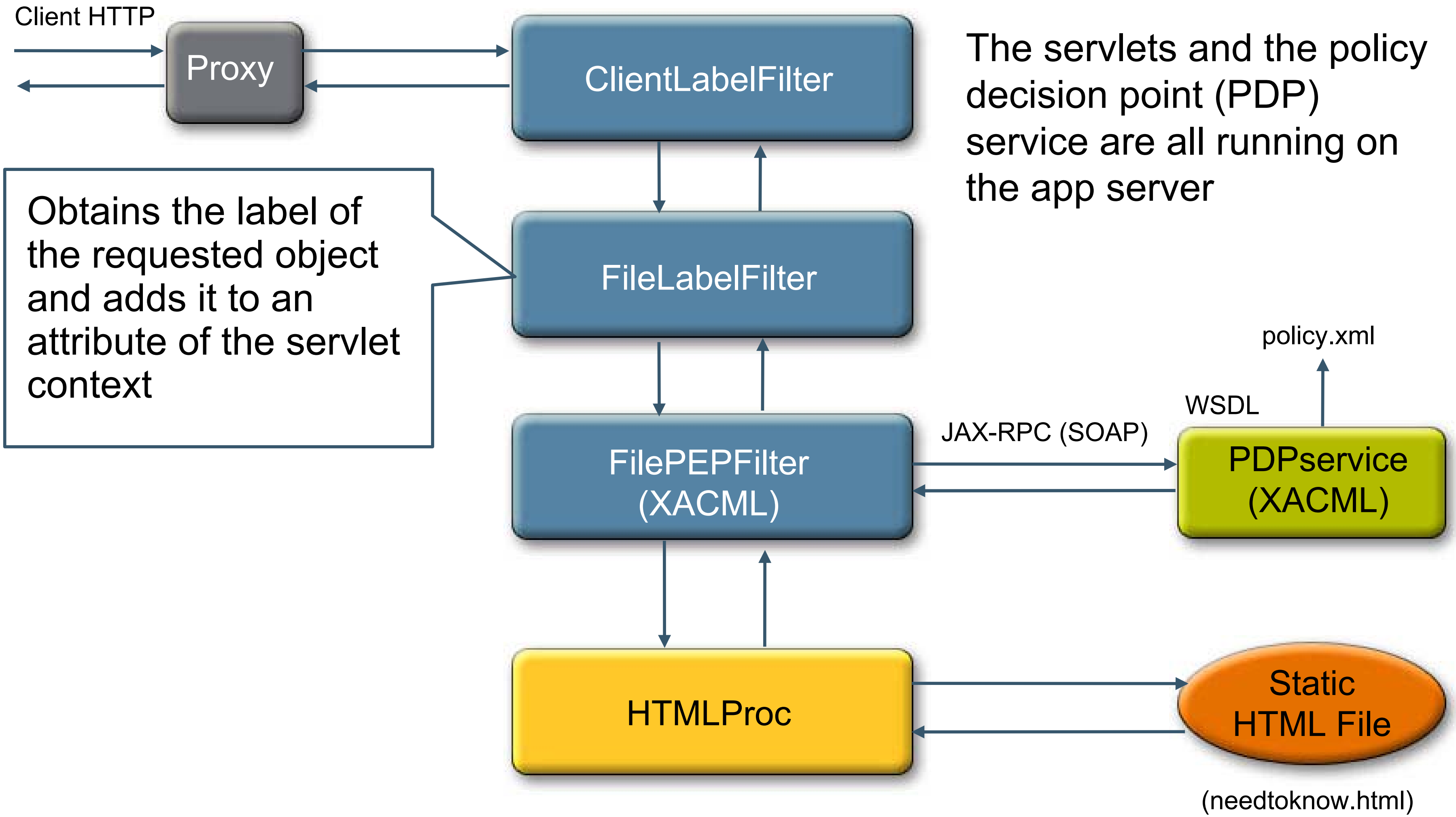


Retrieves connection label from the HTTP field and adds the label as an attribute of the servlet context; If the field is not present, the request is rejected

The servlets and the policy decision point (PDP) service are all running on the app server

JAX-RPC = Java API for XML-based RPC

Prototype Architecture: HTML

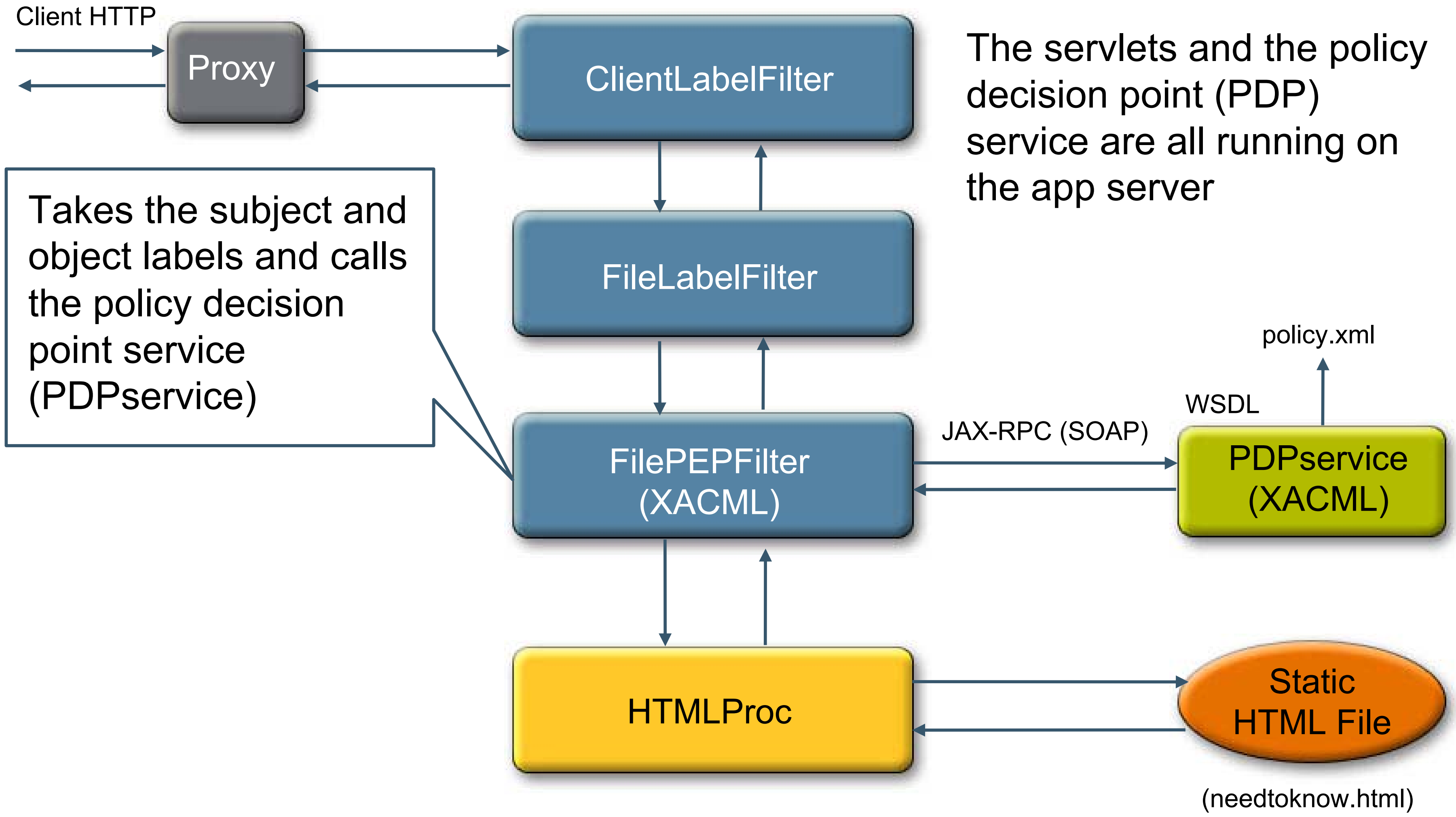


Obtains the label of the requested object and adds it to an attribute of the servlet context

The servlets and the policy decision point (PDP) service are all running on the app server

JAX-RPC = Java API for XML-based RPC

Prototype Architecture: HTML

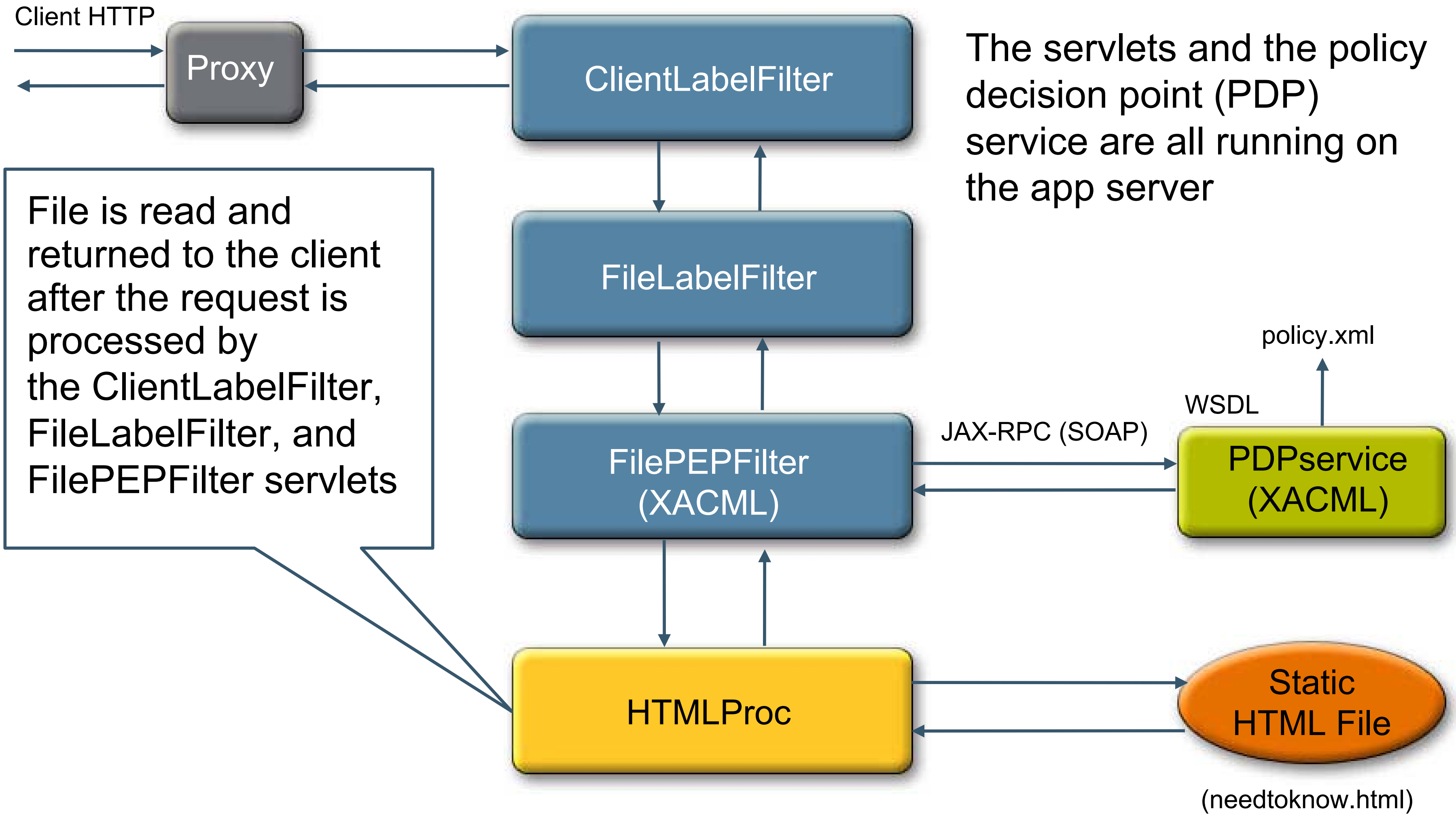


The servlets and the policy decision point (PDP) service are all running on the app server

Takes the subject and object labels and calls the policy decision point service (PDPservice)

JAX-RPC = Java API for XML-based RPC

Prototype Architecture: HTML

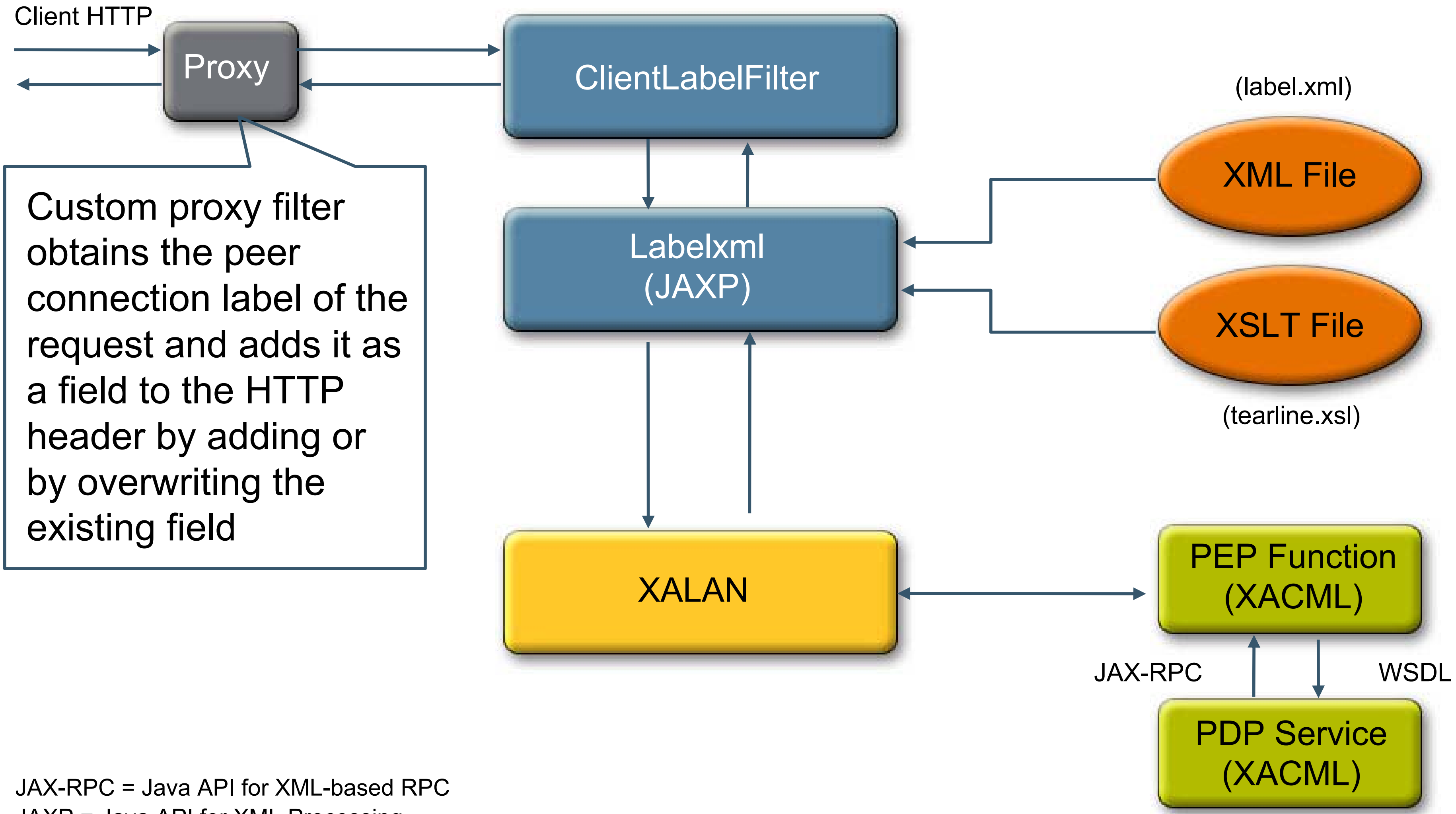


The servlets and the policy decision point (PDP) service are all running on the app server

File is read and returned to the client after the request is processed by the ClientLabelFilter, FileLabelFilter, and FilePEPFilter servlets

JAX-RPC = Java API for XML-based RPC

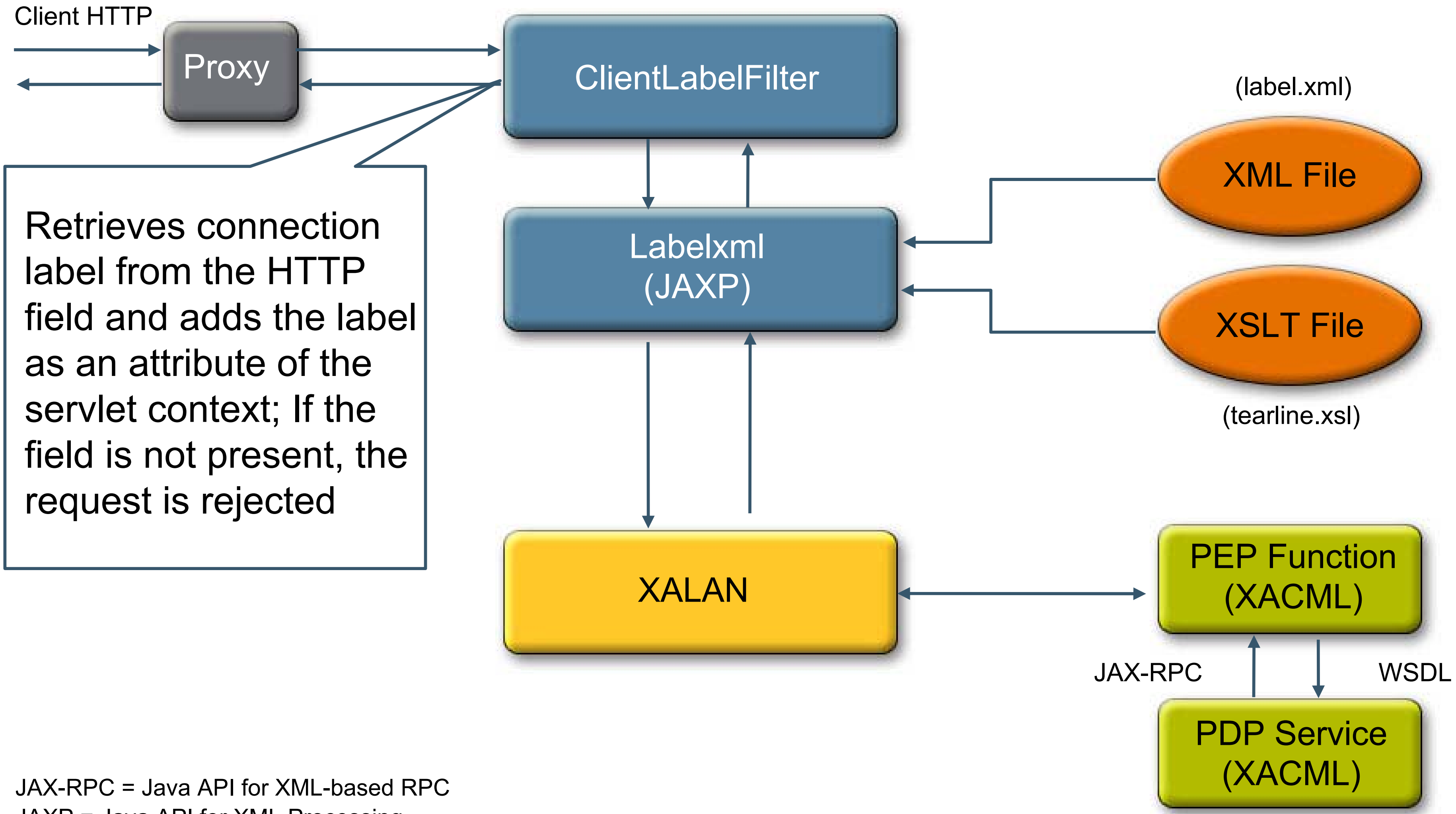
Prototype Architecture: Tearlines XML



Custom proxy filter obtains the peer connection label of the request and adds it as a field to the HTTP header by adding or by overwriting the existing field

JAX-RPC = Java API for XML-based RPC
JAXP = Java API for XML Processing

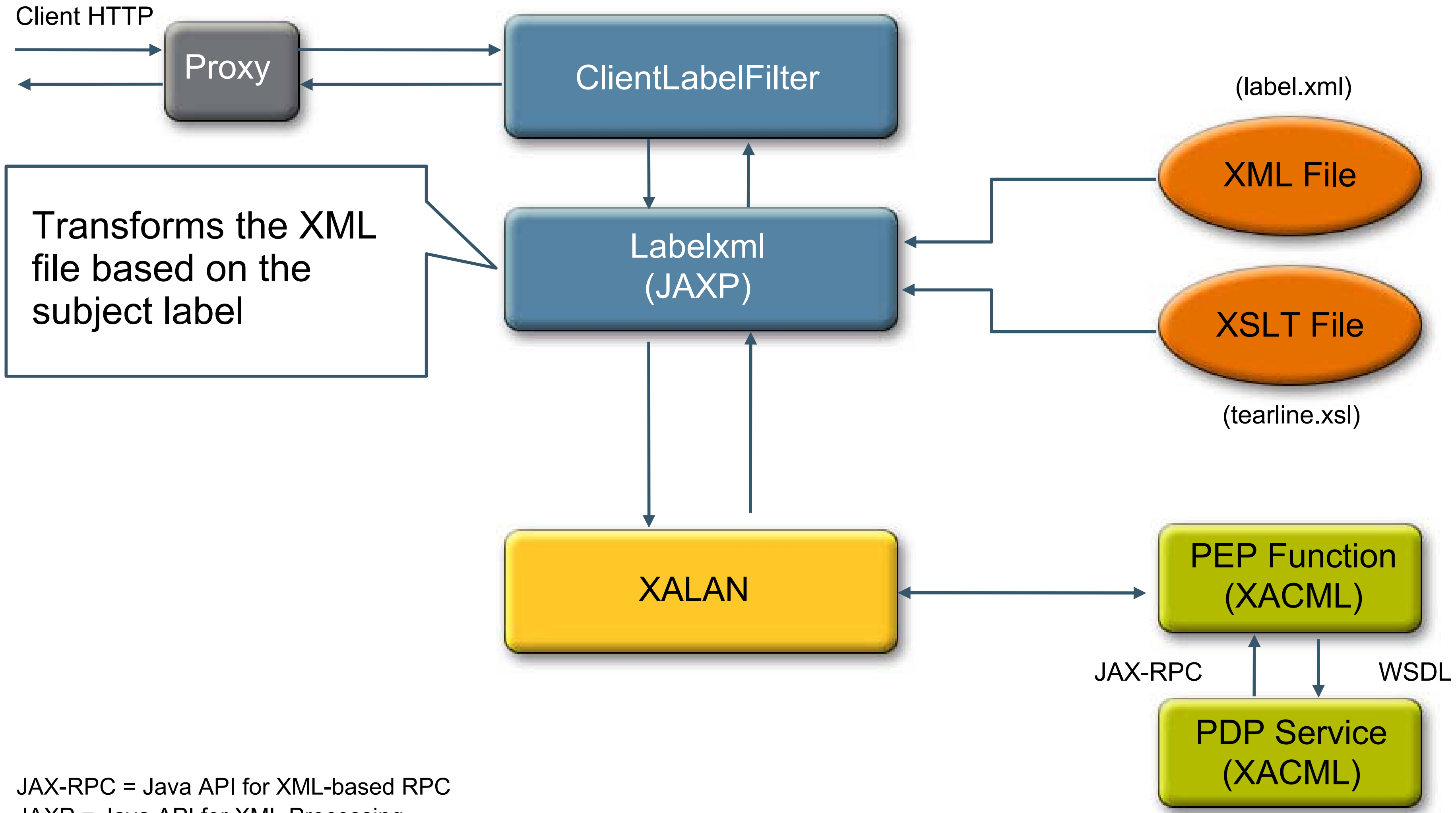
Prototype Architecture: Tearlines XML



Retrieves connection label from the HTTP field and adds the label as an attribute of the servlet context; If the field is not present, the request is rejected

JAX-RPC = Java API for XML-based RPC
JAXP = Java API for XML Processing

Prototype Architecture: Tearlines XML


Client HTTP

Proxy

ClientLabelFilter

Transforms the XML file based on the subject label

Labelxml (JAXP)

(label.xml)

XML File

XSLT File

(tearline.xsl)

XALAN

PEP Function (XACML)

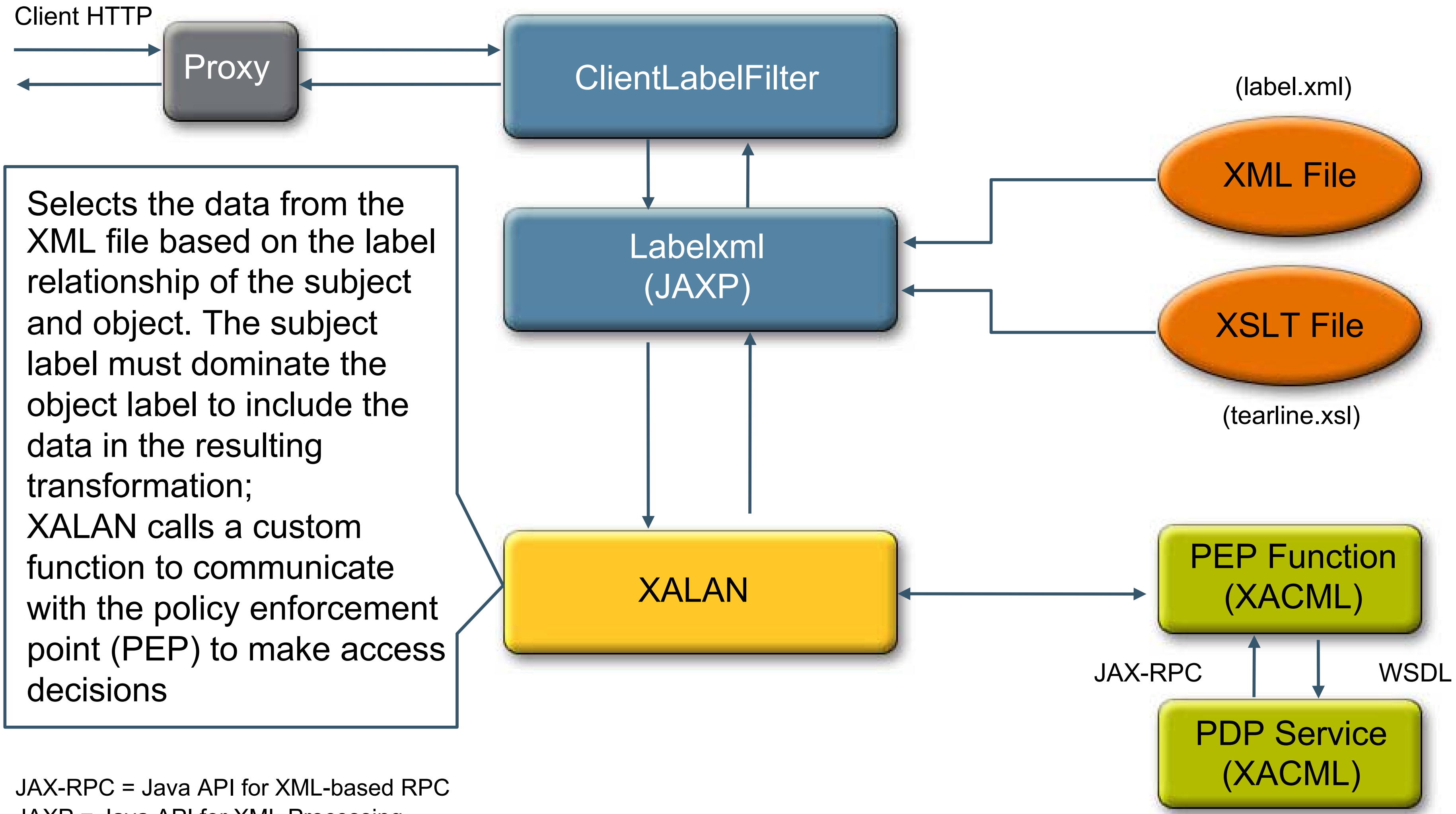
JAX-RPC

WSDL

PDP Service (XACML)

JAX-RPC = Java API for XML-based RPC
JAXP = Java API for XML Processing

Prototype Architecture: Tearlines XML



Selects the data from the XML file based on the label relationship of the subject and object. The subject label must dominate the object label to include the data in the resulting transformation;
XALAN calls a custom function to communicate with the policy enforcement point (PEP) to make access decisions

JAX-RPC = Java API for XML-based RPC
JAXP = Java API for XML Processing

XSLT Transformation File Sample

```
<xsl:template match="document/para">
   <xsl:with-param name="sensitivity_label"/>
   <xsl:if
    test="xsltPEP:dominates($clearance,@sensitivity_label)">
     <xsl:variable name="labelcolor"
      select="label:color(@sensitivity_label)"/>
     <p>
     <xsl:text>(</xsl:text>
     <font color="{$labelcolor}">
     <xsl:value-of select="@sensitivity_label"/>
     </font>
     <xsl:text>)</xsl:text>
     <xsl:apply-templates/>
     </p>
   </xsl:if>
</xsl:template>
```

label.xml File Fragment

```
<para sensitivity_label="CONFIDENTIAL : INTERNAL USE ONLY">
For test purposes only -- If this were a real document, this
paragraph would contain information with a sensitivity label
of "INTERNAL USE ONLY".
<link href="internal.html">Click to see an INTERNAL USE ONLY
HTML page.</link>
</para>

<para sensitivity_label="PUBLIC">
For test purposes only -- If this were a real document, this
paragraph would contain information with a sensitivity label
of "PUBLIC".
<link href="public.html">Click to see an PUBLIC HTML
page.</link>
</para>
```

XACML Request Subject

```
<Subject
 SubjectCategory=”urn:oasis:names:tc:xacml:1.0:
  subjectcategory:access-subject”>
  <Attribute
   AttributeId=”urn:oasis:names:tc:xacml:1.0:subject:
   subject-id”
   DataType=”http://www.w3.org/2001/XMLSchema#string”>
   <AttributeValue>0x0002-08-08</AttributeValue>
 </Attribute>
</Subject>
```

XACML Request Resource

```
<Resource>
  <Attribute
   AttributeId=”urn:oasis:names:tc:xacml:1.0:resource:
    resource-id”
   DataType=”http://www.w3.org/2001/XMLSchema#string”>
    <AttributeValue>0x0002-08-08</AttributeValue>
  </Attribute>
</Resource>
```

XACML Request Action

```
<Action>
  <Attribute
   AttributeId=”urn:oasis:names:tc:xacml:1.0:action:
    action-id”
   DataType=”http://www.w3.org/2001/XMLSchema#string”>
  <AttributeValue>fileread</AttributeValue>
  </Attribute>
</Action>
```

PDP XACML Response

```
[#|2006-03-16T20:11:40.477-0800|INFO|sun-appserver-
pe9.0|javax.enterprise.system
.container.web|_ThreadID=13;_ThreadName=httpWorkerThread-
8080-2;|WebModule[]JFilePEPFilter: PDPservice xacxml
response
<Response>
  <Result ResourceID="0x0002-08-08">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode
        Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

PDP Policy: fileread Action

```
<Actions>
 <Action>
  <ActionMatch
   MatchId="urn:oasis:names:tc:xacml:1.0:function:
   string-equal">
   <AttributeValue
    DataType="http://www.w3.org/2001/XMLSchema#string">
    fileread</AttributeValue>
   <ActionAttributeDesignator
    DataType=”http://www.w3.org/2001/XMLSchema#string"
    AttributeId="urn:oasis:names:tc:xacml:1.0:action:
    action-id"/>
  </ActionMatch>
 </Action>
</Actions>
```

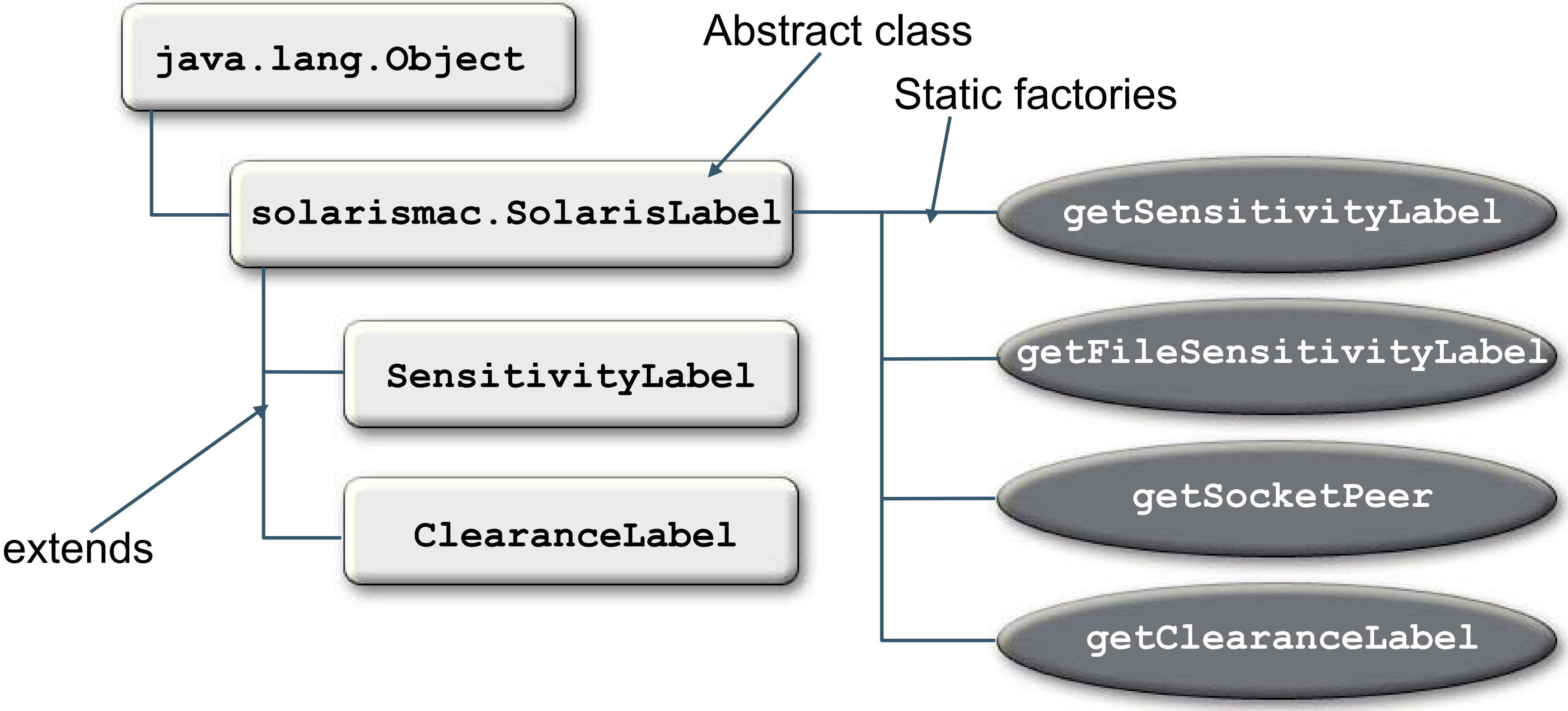
PDP Policy: fileread Rule Fragment

```
<Rule RuleId="FileRead" Effect="Permit">
 <Condition
  FunctionId=”http://
  research.sun.com/projects/xacml/names/function#label-dominates”>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
   string-one-and-only">
   <SubjectAttributeDesignator
    DataType="http://www.w3.org/2001/XMLSchema#string"
    AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"/>
  </Apply>
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
   string-one-and-only">
   <ResourceAttributeDesignator
    DataType="http://www.w3.org/2001/XMLSchema#string"
    AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
  </Apply>
...
```

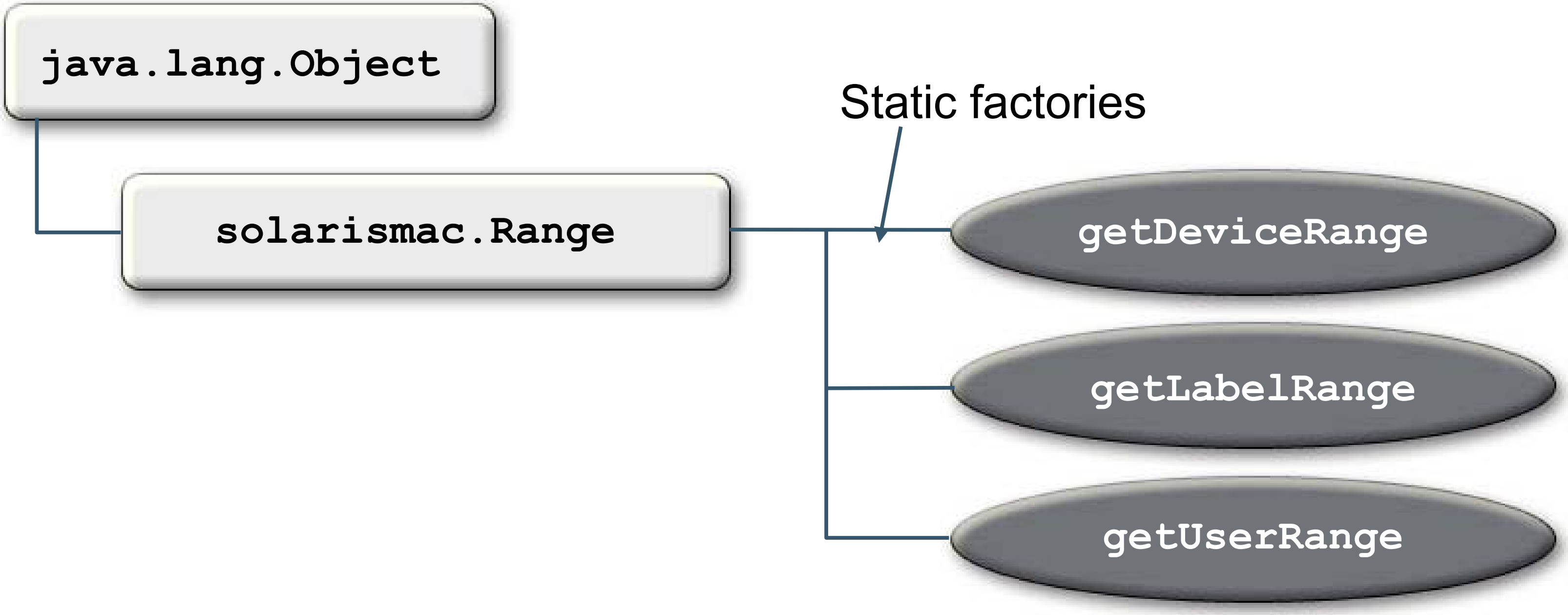
Java Code Bindings for Solaris OS

- Mirror Trusted Extensions label APIs for C
- Uses Java Native Interface (JNI™) to call OS interfaces
- Bindings designed for Java code programmers in mind
  - No mere “port” of APIs from C to Java technology
  - Follows Java code conventions, not C conventions
  - Takes advantage of Java programming language features
    - Static factories
    - Strongly typed
    - Automatic garbage collection
  - Follows Java technology standards

Java Code Bindings for Solaris OS— Classes



Java Code Bindings for Solaris OS— Classes



SolarisLabel Abstract Class

- Includes several general-purpose methods for comparing labels and retrieving data from labels
  - Comparison—dominates(), equals(), strictlyDominates()
  - Bounding—getMinimum(), getMaximum()
  - Presentation—toColor(), toText(), toTextLong(), toTextShort(), toString()
  - Representation—toInternal()

Java Code Bindings—Labeled Printing

- SensitivityLabel subclass includes methods for multilevel printing
  - toHeader(), toFooter(), toProtectAs(), toCaveats(), toChannels()

```
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
              TOP SECRET
This output must be protected as:
TOP SECRET A B SA
unless manually reviewed and downgraded.
  (FULL SA NAME)
  HANDLE VIA (CH B)/(CH A) CHANNELS JOINTLY
              TOP SECRET
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Java Code Bindings—Range Class

- Includes general-purpose methods
  - getLower(), getUpper()
  - inRange()

Still More Areas to Explore

- Audit Class
  - Enable a label-aware application to insert audit records into the Solaris OS audit stream
- Java Security Extension Manager Hooks
  - Add framework to enable the JNI specification code to call the Java Security Extension Manager
- SELinux
  - Expand the prototype to encompass SELinux and its MAC implementation called the Type Enforcement model
- Datagrams (UDP)
  - Determine the sensitivity label of sent and received DatagramPacket objects

Now It’s Your Turn!

- You are a valuable resource in the MLS arena
- Great potential for MLS web services
  - For governmental uses
  - For commercial uses (health care and financial)
- Visit the Trusted Extensions page at OpenSolaris.org
- Install Trusted Extensions on your laptop
- Download the Java code bindings tarball
- Learn about Trusted Extensions
- Write your own label-aware web services
- Tell us what you’ve been doing and how to help

For More Information

- John Weeks' Blog—http://blogs.sun.com/johnw/
- Solaris Trusted Extensions project on OpenSolaris.org— http://www.opensolaris.org/os/community/security/projects/tx/
- Java code bindings tarball
- Servlet-based examples tarball
- Glenn Faden: Trusted Blogger— http://blogs.sun.com/gfaden/
- Trusted Extensions and SELinux Comparison— http://www.sun.com/bigadmin/features/hub_articles/mls_trusted_exts.jsp

For More Information

- Solaris Trusted Extensions Documentation—http://docs.sun.com/app/docs/coll/175.12
- Java Native Interface Documentation—http://java.sun.com/j2se/1.5.0/docs/guide/jni/
- XACML—http://sunxacml.souceforge.net/
- SELinux—http://www.nsa.gov/selinux/

Q&A

Leveraging Solaris Trusted Extensions to Implement Platform Security Services for the Java™ Language

John Weeks

Senior Staff Engineer
Sun Microsystems Federal, Inc.
http://blogs.sun.com/johnw/

TS-1427