

# EPAL to P3P Converter Design Document

Version 1.5

Nov 15, 2003

Pei-Chao Weng  
Kenneth Chu

# Table of Contents

<b>0.</b>	<b>Abstract</b>	<b>3</b>
<b>1.</b>	<b>Introduction</b>	<b>3</b>
<b>2.</b>	<b>Assumptions</b>	<b>4</b>
<b>3.</b>	<b>Constraints</b>	<b>4</b>
<b>4.</b>	<b>Design Philosophy</b>	<b>4</b>
<b>5.</b>	<b>System Overview</b>	<b>5</b>
	5.1 Mapping Functionality	
	5.2 Transformation Functionality	
	5.3 Development Choice	
<b>6.</b>	<b>Dataflow</b>	<b>7</b>
<b>7.</b>	<b>Element-Specific Details</b>	<b>8</b>
<b>8.</b>	<b>Global Data and File Structure</b>	<b>14</b>
	8.1 EPAL Vocabulary	
	8.2 Business Intelligence	
	8.3 P3P Mapping Info	
	8.4 EPAL Policy	
	8.5 P3P Statement	
<b>9.</b>	<b>UI and API</b>	<b>15</b>
<b>10.</b>	<b>Expected Schedule</b>	<b>15</b>
<b>11.</b>	<b>Document Revise History</b>	<b>16</b>
<b>12.</b>	<b>Appendix</b>	<b>16</b>

# Abstract

Enterprises advertise their privacy policies using P3P (Platform for Privacy Preference), which defines what recipients can obtain what collected data from web users for what purposes. Internally, enterprises can also define fine-grained privacy practices using EPAL (Enterprise Privacy Authorization Language). These internal policies should reflect and guarantee the promises to their customers. Because the internal privacy policies can change frequently, it can be challenging to keep the internal privacy policies in sync with outstanding promises. As a consequence, we deliver a system which can automatically translate internal EPAL policy into outstanding P3P privacy statement.

## 1. Introduction

W3C has already recommended **P3P (Platform for Privacy Preference)** as tools to inform web users of the data collection practices of the websites. Thus, enterprises can publish privacy promises with P3P. It defines what recipients can obtain what collected data for what purpose. In brief, P3P provides a way for websites to encode its data-collection and –uses practices in a machine-readable XML format, in order to inform enterprise websites’ customers or visitors about privacy issues.

Internally, enterprises can use **EPAL (Enterprise Privacy Authorization Language)** to define and enforce privacy-related policies. EPAL has been proposed by IBM. It defines an interoperability language for exchanging privacy policy in a structured format between applications or enterprises. In brief, EPAL is defined by an XML schema since the privacy policies can easily be described and enforced extensively and effectively.

Ideally, enterprises’ internal privacy policies should guarantee the effectiveness of their promises made to their customers. If there are something changes on these policies, the customers must be informed and updated if needed. Now this sync-privacy-policy job can only be done manually. It’s a little troublesome and can easily go astray, making the inconsistency between the internal and outstanding policies.

It would be good if there is automatically mechanism to keep the outstanding promises (P3P policies) up-to-date with internal practices (EPAL policies). The purpose of the project is to develop a converter, which can automatically translate EPAL policies to corresponding P3P statements. It can help enterprises ensure the consistency and persistence of their privacy-related policies.

## 2. Assumptions

Attributes mapping assumption is used in this system in order to deal with ambiguities in P3P specification. The assumptions are as the followings.

- Map “EPAL data categories” to “P3P categories and/or elements”. In hierarchical elements, each mapping function labels P3P-relevant leaf elements of the EPAL hierarchy. These elements include “user-category”, “data-category” and “purpose”.
- Retention and deletion. The deletion of the collected information is defined in “retention” element in P3P, and possibly “obligation” element in EPAL. Thus we assume the mapping of the 2 elements.
- P3P-aware EPAL policy editing. To get a better mapping between the two policy statements, EPAL policy editor must have some insights about P3P.

However, in order to fulfill various requirements in different environments, most of these assumptions can be manually changed.

Please refer to appendix for more element-specific assumptions.

## 3. Constraints

- EPAL and P3P may have different semantics and interpretation on data elements. If there is inconsistency between the two standards, the leaf mapping algorithm described above would not be practical. Thus more elegant and difficult mapping algorithm is needed.

Please refer to appendix for more element-specific constraints.

## 4. Design Philosophy

The system can be roughly divided into 2 parts.

1. Mapping Part
2. Translation Part

**Mapping part** is responsible for initializing the transformation phase.

[Usage] This part would be used to generate *P3P mapping info* from existing *EPAL Vocabulary* file. In other words, when there is something change in *EPAL Vocabulary* file, or it's the first time using the system, the “mapping functionality” should be executed first to generate the latest corresponding P3P mapping information.

[Input and Output] This part takes “EPAL definition” and “Business Intelligence” as input, and brings out “P3P mapping info” as output. It extracts EPAL definition to

generate P3P-specific details that cannot be derived from the EPAL policy. Thus, the phase might not be 100% automatic. It may require some manual efforts to better map attributes between the 2 formats (EPAL and P3P). Finally the administrator can decide whether the manually input data would be added into “Business Intelligence” component. The growing “intelligence” would make future mapping more precisely and gradually fully reflect enterprise’s needs.

The mapping part can be used in first-time use and update info in order to be sync with any internal policy changes.

**Translatiønn part** is responsible for converting EPAL policies to P3P policies.

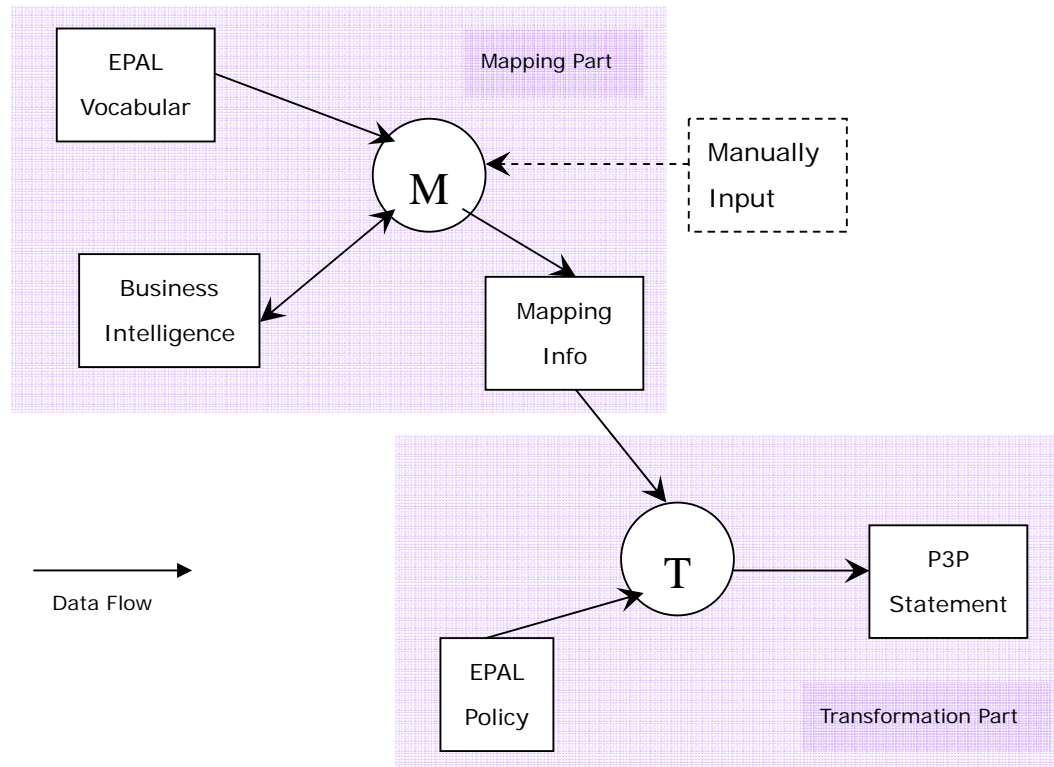
**[Usage]** This part would be used to generate *P3P policy statements* with the combination of *EPAL policy file* and *P3P mapping info* generated from the mapping part.

**[Input and Output]** This part takes “EPAL policy” and “P3P mapping info” as input, and brings out “P3P statement” as output.

The translation part can be used to generate corresponding P3P statements with internal related privacy policies.

## 5. System Overview

The system contains 2 main functionalities, mapping and transformation.



### 5.1 Mapping functionality

It brings in “EPAL Vocabulary” to generate “P3P mapping information” with possible manual assistance. The EPAL Vocabulary info, which comes from the enterprises internal privacy policy definition, is XML format. P3P mapping info and business intelligence info are not necessarily to be in XML. They can be any format, but XML is still recommended for its convenience. Refer to section 8 for more information about data file format.

### **5.1.1 Business Intelligence**

The system would have some “built-in” intelligence, which presets and recommends some mapping between EPAL and P3P. This built-in feature can be seen as a “learning” mechanism of the system. When there is some policy editor’s input, which is different from built-in predefined mapping mechanism, the editor can choose to (or not to) save the input information to the system, then the saved information would be processed into appropriate format, being saved in the system as future reference and speeding process of other EPAL vocabulary files if needed.

For example, the content of *<purpose>* element in EPAL Vocabulary file would be simply translated into *<purpose>* element in P3P statement. But this time the enterprise’s policy editor has the following different assumption in mind: if “Taipei 101 Opening Celebration” appeared in any *<purpose>* hierarchy in EPAL Vocabulary, the corresponding *<policy>* element in P3P statement should be encoded as *<other-purpose> Special Event: Taipei 101 Opening Celebration </other-purpose>*. The policy editor can choose to save the special translation method for future uses to speed the mapping process and reduce manual efforts.

The system also has 2 views to show the mapping function, from EPAL’s view and from P3P’s view. Take EPAL’s view as an example, the GUI panel would show various EPAL vocabulary info, with their corresponding P3P attributes. There would be

1. User category
2. Data category
3. Action
4. Purpose
5. Obligation

The GUI panel can be divided into two sides, with corresponding attribute pairs. Each subpart can be manually changed to better reflect different policy definition within different enterprises.

Business Intelligence is a vocabulary-based learning system rather than element-based.

### **5.2 Translation Functionality**

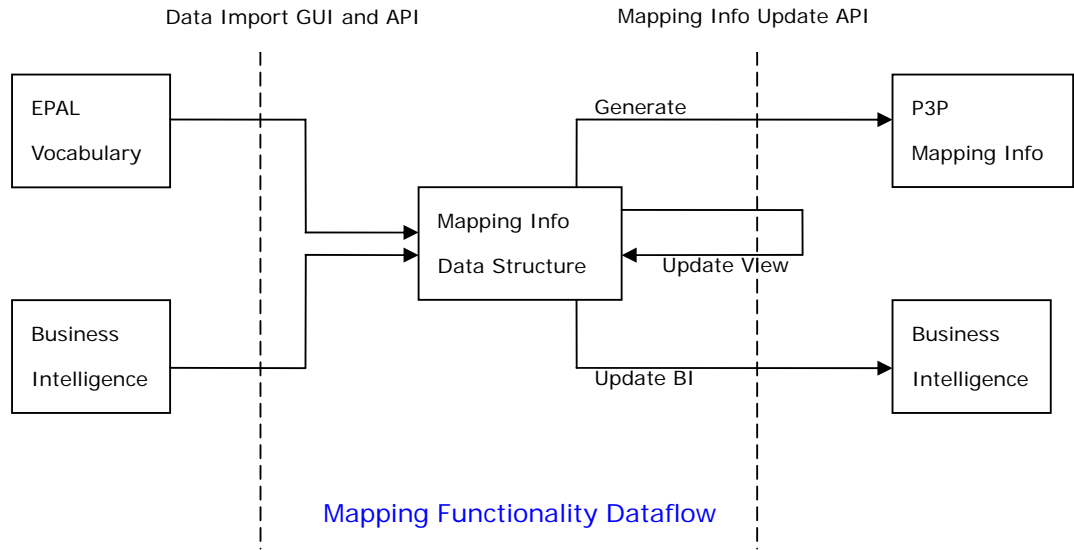
The mapping info would be the input of translation functionality, and with accompanying “EPAL policy”, finally we get “P3P policy”. Both “EPAL policy” and “P3P policy” are XML documents, and each of them adheres to its XML schema defined in specifications, respectively.

### 5.3 Development Choice

We will use JDK V1.4.2 to fully utilize its XML processing capabilities, and it's also the latest version during our development phase.

## 6. Dataflow

### 6.1 Dataflow in Mapping Part



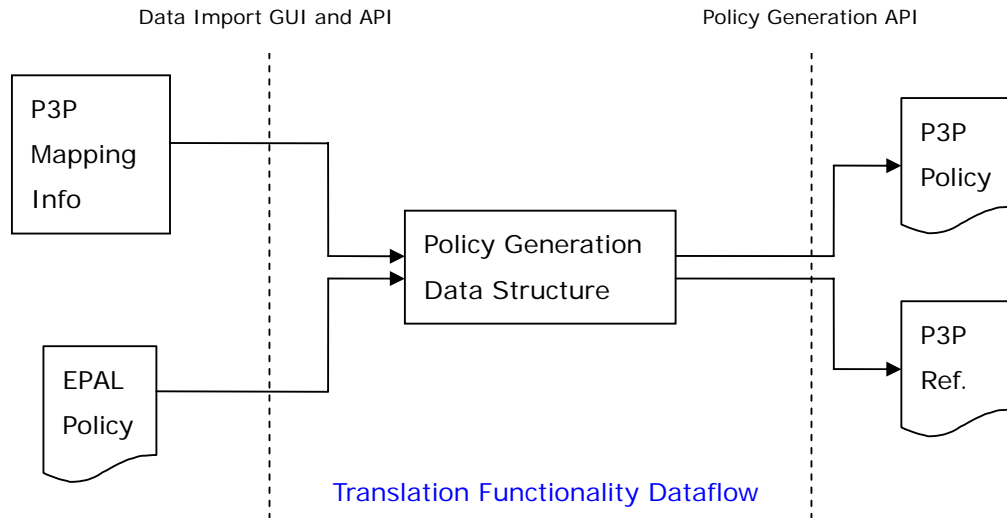
The mapping part consists two sets of API, for data import and info update, respectively. As the figure above, **Data Import GUI and API** is responsible for reading EPAL vocabulary info and pre-defined mapping info (Business Intelligence) to fill in mapping info data structure.

After the initialization phase, the other set of API, **Mapping Info Update API**, would be used. Mapping info data structure would be displayed on a panel, showing the relationship between corresponding elements of EPAL and P3P. The policy editor can update the predefined mapping manually on that panel, and the changes would be reflected on the mapping info data structure on the fly (Update View).

Finally, the policy editor can choose to generate P3P mapping info, or update the “intelligence” by the current policy editor, or both.

### 6.2 Dataflow in Translation Part

The translation part also contains two sets of API, for data import and policy generation. As the figure below, **Data Import API** brings in *P3P mapping info* and *EPAL policy file*, then fill the internal data structure. The preview of the translation would be shown on the panel, according to the P3P mapping info file. Once the policy editor is satisfied with the translation preview, the **Policy Generation API** would simply writes P3P data to physical policy file and policy reference file.



## 7. Element-Specific Details

### 7.1 EPAL Rule

Rules are the core elements of an EPAL document, while statements are the core elements of a P3P documents. Therefore it is natural to map rules to statements.

#### 7.1.1 Assumptions

A rule may have a ruling of “allow” or “deny”. From a user’s point of view, who is trying to protect his or her private data, what the enterprise will do to the data is more important than what the enterprise will not do. In addition, deny rules are usually for internal reasons, which the users do not need to know. And the default ruling is deny. Due to the three reasons above, we do not translate deny-rules.

#### 7.1.2 Constraints

Since a rule is a collection of Data-Categories, User-Categories, Purposes, Actions, Conditions, and Obligations, the constraints of rules are collections of constraints of all the subelements.

#### 7.1.3 Mapping Info

There is no mapping info for rule because rules are not subelements of epal-vocabulary.

### 7.2 User-Category and Purpose

User categories and purpose are defined hierarchically in epal-vocabulary, and they have perfectly matching counterpart in P3P, recipient and purpose, respectively.

#### 7.2.1 Assumptions

We require a predefined *UserMap* and *PurposeMap* to gain knowledge about the

relationships between (User-category and Purpose) and (Recipient and Purpose). The same assumption applies to Data-Category, Obligation, and Condition. We also assume that all the describedObjectType elements in the epal-vocabulary have their unique id's.

### **7.2.2 Constraints**

We may need manual assistance or proper Business Intelligence to determine the value. This applies to other elements as well.

### **7.2.3 Mapping Info / Business Intelligence Entry**

List of mapping from id's of the EPAL source node to P3P values, e.g.

An entry in the *UserMap*: (r-and-d, <ours/>)

An entry in the *PurposeMap*: (delivery, <current/>)

It is not necessary to remember to full path of an EPAL element in its hierarchical structure (e.g. all/internal/r-and-d/) because all describeObjectType elements have unique id's.

The right hand side of an entry of both element may have multiple values. Purpose may have self-defined P3P values, while User-category may not.

### **7.2.4 Pseudocode**

See 7.3.4

## **7.3 Data-Category**

Data-Category shares the same properties as User-Category and Purpose, with one additional requirement. Because P3P statements may optionally contain <NON-IDENTIFIABLE/> and no other elements, we want to express that when appropriate.

### **7.3.1 Assumptions**

See 7.2.1

### **7.3.2 Constraints**

See 7.2.2

### **7.3.3 Mapping Info / Business Intelligence Entry**

There are two mappings associated with Data-Category, *DataCategoryMap* and *IdentMap*. *DataCategoryMap* maps nodes of EPAL Data-Category hierarchy to predefined P3P data schema or a wrapper data schema #dynamic.miscdata with detail categories, and *IdentMap* maps nodes of EPAL Data-Category to either Non-Ident or Ident. e.g.

*DataCategoryMap*:

(all/customer/id, #user.login) – predefined P3P data schema

(all/customer/religion, #dynamic.miscdata

<categories><demographic/></categories>)

*IdentMap*:

(all/customer/id, Ident)

(all/customer/religion, Non-Ident)

Note: the full path does not exist in mapping info as stated in 7.2.3. The inclusion of full path is just for demonstration.

Exactly one data schema must be chosen for one entry. If the data schema is #dynamic.cookies or #dynamic.miscdata, category must not be empty, otherwise, category must be empty.

#### **7.3.4 Pseudocode**

if (all the data are non-ident according to *IdentMap*)

```
<statement><non-identifiable/></statement>
```

else

```
<statement>
```

```
<purpose>
```

unionize all the involved *PurposeMap*.P3P-element here

```
</purpose>
```

```
<recipient>
```

unionize all the involved *UserMap*.P3P-element here

```
</recipient>
```

```
<data-group>
```

for each data-category involved

```
<data ref="DataCategoryMap.data-schema">
```

(optional) <categories>...</categories>

```
</data-group>
```

```
</statement>
```

### **7.4 Obligation, Action, and Retention**

The reason for data retention can be interpreted from EPAL obligation and action elements. If no data is stored, then the retention is no-retention. Otherwise, we have to determine the retention from obligation.

#### **7.4.1 Assumptions**

We assume each obligation, identified by their id's, has a unique retention-purpose, although the same obligation could be reused for different retention purposes.

#### **7.4.2 Constraints**

P3P does not have a proper element for action or obligation. The mapping from action and obligation to retention is the closest match.

#### **7.4.3 Mapping Info / Business Intelligence Entry**

*ActionMap* maps EPAL actions to one of the following five values: create, update, disclose, use, and delete.

*RetentionMap* maps EPAL sets of obligations under one rule to (retention purpose, a

human-readable-explanation).

Note that the structure (left side of the mapping) of *RetentionMap* is different from other maps. The reason is that there could be more than one obligation under one EPAL rule and only one retention under one P3P statement. Therefore instead of mapping one obligation to one retention, we map sets of obligation to one retention.

#### **7.4.4 Pseudocode**

```
place the action in <consequence>
if (one of the actions is disclose)
    retention = indefinitely
else if (one of the actions is create or update)
    if(least friendly retention of all the obligations== indefinitely)
        // least friendly = indefinitely
        // medium friendly = legal, business, or stated
        // most friendly = no-retention
        retention = indefinitely
    else if (least friendly retention of all the obligations== medium friendly)
        for each distinct retention
            create an identical statement except the retention part
            place the RetentionMap.human-readable-explanation in the file discuri
            references to
        else
            retention = no-retention
    else // use and delete
        retention = no-retention
```

#### **7.5 Condition**

The translation of Conditions is the least intuitive. The resulting P3P counterpart is scattered across the P3P documents, as explained later in the Mapping Info and pseudocode.

##### **7.5.1 Assumptions**

P3P only has the mechanism to indicate opt-in or opt-out. We ignore all other condition. By ignoring them, it is equivalent as assuming they always return true under allow-rule. Consider the following example: “we look at your data if you are over 18”. Ideally, we would present a P3P statement “we look at you data” to users who are over 18, and no statement to users who are under 18. But that is not possible. By assuming the condition always return true, we present the same statement to all users. In some sense, the statement can be interpret as “we MIGHT look at your data”. In fact, this assumption does not violate the promising of privacy protection.

We further assumes that there is no contradicting Conditions under a rule.

### **7.5.2 Constraints**

Conditions do not fit anywhere in the P3P semantics, therefore we must use extension to represent condition in P3P. Because both XACML conditions and P3P extensions are in XML format, one way to transfer Condition is just copy the entire Condition clause and the associated Containers and paste it under the extension. The method is easy and does not lose information. A P3P agent with minimal functions may not understand the extension, but this is the only way to preserve information content. We can only hope the P3P agent understands XACML conditions.

### **7.5.3 Mapping Info**

*OptMap* maps Conditions (id's) to ({opt-in-cond, opt-out-cond, none}, instruction on how to opt-in/out in the future).

### **7.5.4 Business Intelligence Entry**

*OptMap* is not part of BI Entry because Conditions are defined in EPAL policies rather than vocabularies.

### **7.5.5 Pseudocode**

```
<statement>
  <extension>
    <containers>
      Place the associated containers here
    <condition>
      Place the original code here
  </extension>
</statement>
```

```
if (condition is an opt-in-cond)
  add "required=opt-in" attribute to all the translated P3P purposes
else if (condition is an opt-out-cond)
  add "required=opt-out" attribute to all the translated P3P purposes
else
  do nothing // see 7.5.1 Assumption
```

```
if (condition is an opt-in-cond or an opt-out-cond)
  Place the how-to instruction in the opturi file
  <data optional="yes">
```

```
if(there are exist an opt-in condition and an opt-out condition)
  duplicate the statement, one with "required=opt-in", the other with
  "required=opt-out"
```

## **7.6 Access**

### 7.6.1 Assumptions

We assume the input EPAL policy (an internal access control) is the defining access control document for an external web user as well.

### 7.5.2 Constraints

Under P3P specification, there can be only one access under a policy. Therefore, any information associated in EPAL will be reduced after translation.

### 7.5.3 Mapping Info / Business Intelligence Entry

AccessSubjectMap: (user-category, true/false)

True mapping value represents the fact that this user category is an access subject.

AccessPurposeMap: (EPAL purpose, true/false)

True mapping value represents the fact that this purpose is an access purpose.

AccessAllMap: (data-category, true/false)

True mapping value represents the fact that this data category is of access all type.

AccessContactMap: (data-category, true/false)

True mapping value represents the fact that this data category is of access contact type.

AccessOtherMap: (data-category, true/false)

True mapping value represents the fact that this data category is of access other type.

### 7.5.4 Pseudocode

```
if(all the rules are not identifiable according to IdentMap)
```

```
    access = nonident
```

```
else if (exists a rule where (any of the purposes ∈ access purposes && any of the user-categories ∈ access subject && any of the actions ∈ use or update && any of data-categories ∈ access all))
```

```
    access = all
```

```
else if (... && any of data-categories ∈ access contact && any of data-categories ∈ access other))
```

```
    access = contact-and-other
```

```
else if (... && any of data-categories ∈ access contact))
```

```
    access = ident-contact
```

```
else if (... && any of data-categories ∈ access other))
```

```
    access = other
```

```
else
```

```
    access = none
```

### 7.7 Miscellaneous

- <property> element in EPAL describedObjectType is EPAL-implementation-specific. Therefore, it can be safely ignored for this document-to-document translation

## 8. Global Data and File Structure

The system has 5 files associated with it. They are "EPAL Vocabulary", "Business Intelligence", "P3P mapping info", "EPAL policy" and "P3P statement". Their functionalities have already been described above. Their XML file format and naming convention are defined below.

### 8.1 EPAL Vocabulary

It would be located in \$USER\_HOME/EPAL\_P3P\_Converter/Vocabulary/

Its XML format

```
<epal-vocabulary>
  <vocabulary-information>....</vocabulary-information>
  <data-user>...</data-user>
  <data-category>...</data-category>
  <purpose>...</purpose>
  <action>...</action>
  <container>...</container>
  <obligation>...</obligation>
  <version>...</version>
</epal-vocabulary>
```

### 8.2 Business Intelligence

It would be located in \$USER\_HOME/EPAL\_P3P\_Converter/BI/

Its XML format

```
<Business_Intelligence>
  <Business_Intelligence_Entry>
    <UserMap>...</UserMap>
    <DataMap>...</DataMap>
    <PurposeMap>...</PurposeMap>
    <RetentionMap>...</RetentionMap>
    <ActionMap>...</ActionMap>
    <AccessMap>...</AccessMap>
    <NonIdentMap>...</NonIdentMap>
  </Business_Intelligence_Entry>
</Business_Intelligence>
```

### 8.3 P3P Mapping Info

A mapping info would be the equivalent of a Business Intelligence Entry plus OptMap. Mapping infos resides in memory only rather than files.

### 8.4 EPAL Policy

It would be located in \$USER\_HOME/EPAL\_P3P\_Converter/EPALPolicy/

Its XML format

```
<epal-policy>
  global-condition
  version
  default-ruling
```

```

    <policy-information>.... </policy-information>
    <epal-vocabulary-info>... </epal-vocabulary-info>
    <condition>... </condition>
    <rule>... </rule>
  </epal-policy>

```

## 8.5 P3P Statement

It would be located in \$USER\_HOME/EPAL\_P3P\_Converter/P3PStatement/

Its XML format

```

  <policies>
    <policy>
      <entity>
        <data-group>
          <data>... </data>
        </data-group>
      </entity>
      <access>
      </access>
      <statement>
        <extension>... </extension>
        <purpose>... </purpose>
        <recipient>... </recipient>
        <retention>... </retention>
        <data-group>
          <data>... </data>
        </data-group>
        <consequence>... </consequence>
      </statement>
      <dispute-group>
        <disputes>
          <remedies>... </remedies>
        </disputes>
      </dispute-group>
    </policy>
  </policies>

```

## 9. UI and API

See Javadoc.

## 10. Expected Schedule

- |                               |               |
|-------------------------------|---------------|
| 1. Literature Survey          | 9/10 ~ 9/24   |
| 2. Conversion Mechanism       | 9/25 ~ 10/10  |
| 3. System Analysis and Design | 10/11 ~ 10/22 |
| 4. System Development         | 10/23 ~ 11/23 |

- 5. Quality Assurance and Testing 11/24 ~ 11/30
- 6. Final Testing and Documentation 12/1 ~ 12/15

## 11. Document Revision History

---

Version	Authors	Date	Description
0.8	Pei-Chao Weng	Oct 13, 2003	Initial Version
1.0	Pei-Chao Weng	Oct 20, 2003	1. More on Business Intelligence 2. Add "detail translation" as appendix 3. Add "abstract" and "expected schedule" 4. Revise some word uses
1.2	Kenneth Chu	Oct 25, 2003	1. Add Section 7 Element-Specific Details 2. Revision of section 9
1.5	Kenneth Chu	Nov 15, 2003	Section 7, 8, and 9 revised.

---

## 12. Appendix

Please refer to another document for detail translation between elements in P3P and EPAL.