W3C Customer Experience Data Group
Privacy Policy Implementation Options - Summary

Murray Williams, Accenture Interactive

The purpose of this document is both to present (and present arguments in favor of) the "Centralized" approach to implementing privacy policy data, and to give some points comparing it with the other two approaches that were mentioned on the June 6 call.

Centralized:

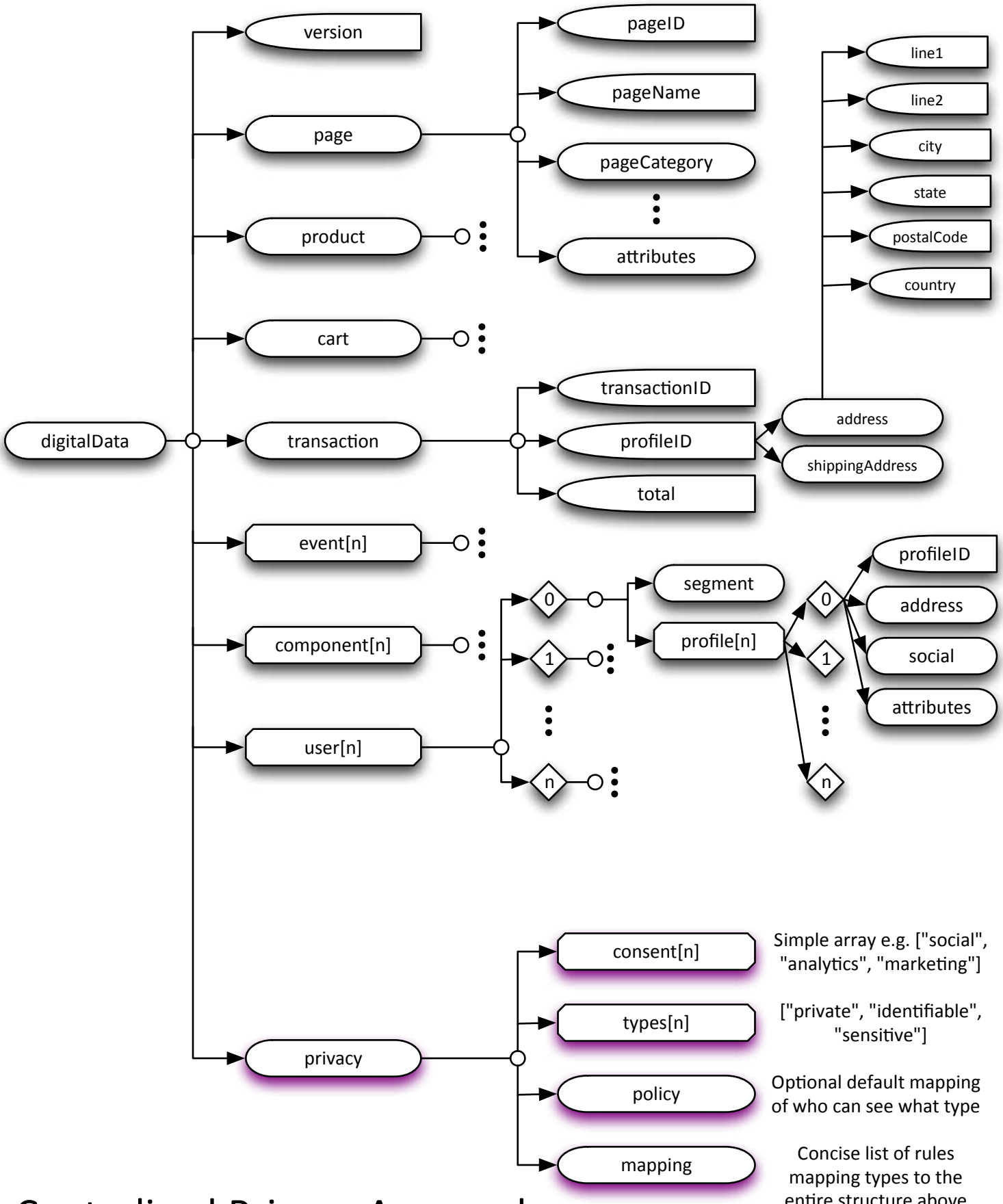
- All information relating to privacy policy is kept under a "privacy" object off of the top level of the tree.
- Hierarchical inheritance is used whenever possible. I.e. set overall policies on the biggest branches and only specify policy on smaller branches or end-nodes when an exception is needed.
- The actual "mapping" of privacy/consent types to the digitalData tree is done by a compact set of "rules" very much like CSS stylesheet rules.

Annotated:

- All information relating to privacy policy is kept under each "branch" of the digitalData tree (e.g. digitalData.page or digitalData.product.productCategory) via a object named "privacy" (e.g. digitalData.page.privacy or digitalData.product.productCategory.privacy).
- If an item requiring a policy annotation is a JavaScript primitive (e.g. a string or number) then the privacy information will be placed in a sibling object with the name *-privacy. So digitalData.transaction.total would have privacy data set under an object called digitalData.transaction.total-privacy.
- All of the "privacy" or "*-privacy" objects have (i) a privacy.types structure that describes which types of privacy apply to the annotated object, and (ii) a privacy.consent structure that states which outside systems (analytics, etc.) are allowed to read the parent annotated object.
- Hierarchical inheritance is NOT allowed, as this requires that the digitalData tree be reverse-navigated to determine if an inherited value should be applied.

Hybrid:

- Same as annotated approach, but end-nodes / primitives are not annotated with a paired sibling "*-privacy" object. Instead, privacy is always kept at the object/list/branch level. Inside the privacy object is a mapping of the ".types" and ".consent" information to all of that branch's end-nodes.



Centralized Privacy Approach

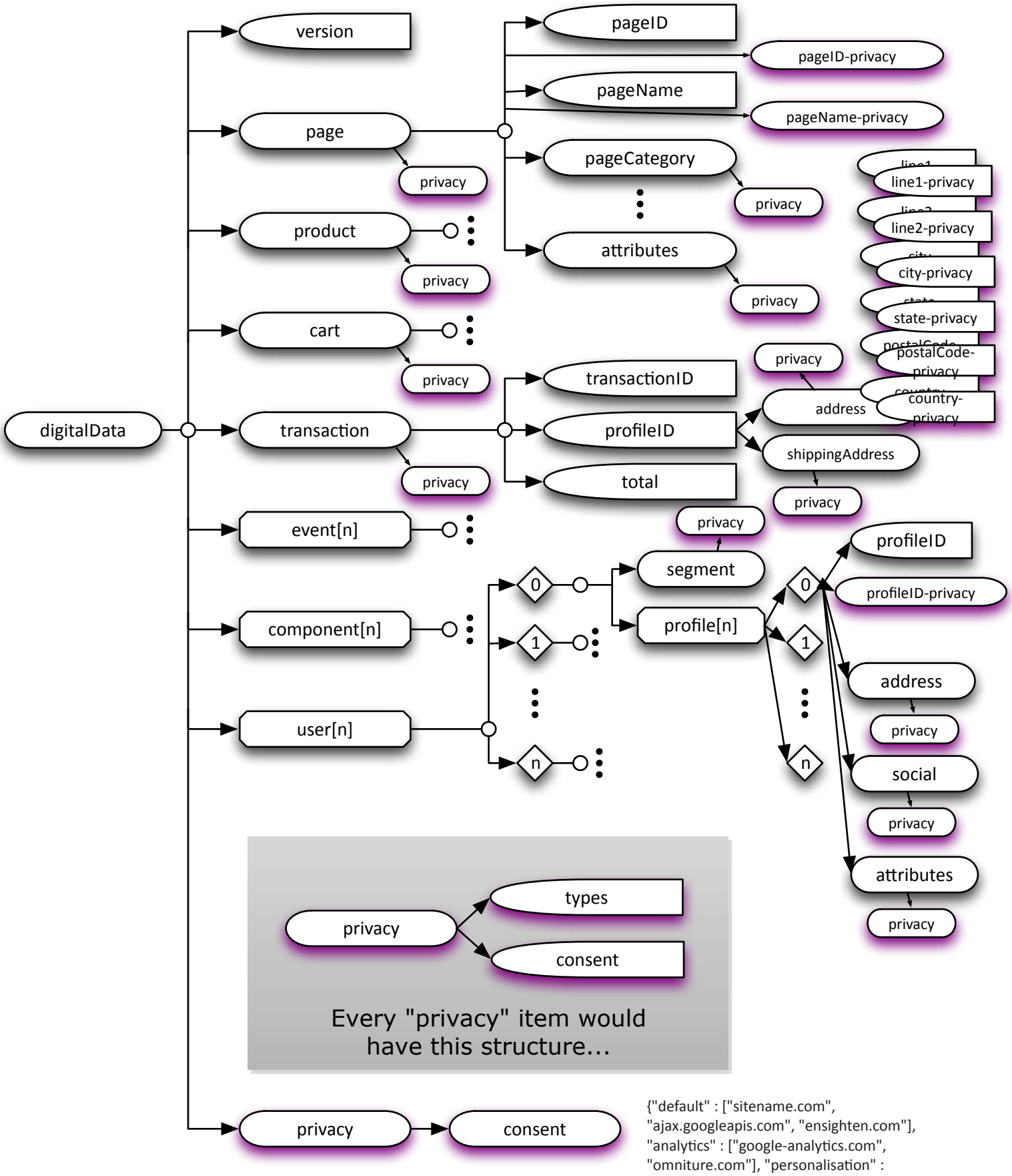
Properties of the Centralized Approach		
Impact on core data	Negligible	All privacy meta-data is kept in a single structure off the top level.
Verbosity	Concise/ Minimal	Design was kept with succinctness in mind. A sufficient privacy policy could be written with as few as five or sex rules.
Complexity	Medium	The "key" for each rule describes a path and has a grammar that is easy to understand, but requires some degree of parsing.
Writing Privacy / Implementation	Easy	Because of its brevity, the entire policy mapping of any page can be done by hand up-front. In the rare case an dynamic exception is needed, it can be appended to the list of rules very easily.
Reading Privacy / Implementation	Moderate*	If the policy is simply being reviewed by a human (i.e. for legal statement) it is succinct and straightforward. But if computer code needs to determine (read) the privacy of any single item in the digitalData structure, some parsing and logic will be required.
Enforceability	Medium/High	This system was designed to be succinct and complete. By explicitly defining the privacy types, consent types, (optional) privacy/consent defaults and privacy/consent mapping rules in one place, software library could be employed to apply an enforcement layer.

Commentary:

When I worked through a couple examples, from the simple product page to the much more complex shopping cart, I realized that the actual privacy rules are:

- **generally static**—you really can hand-write them in one go. In fact, I believe a privacy implementer is going to think very carefully once about what parts of the page have what privacy, and they are going to want to write it down in one place.
- **straightforward**—page, product, event, and components display public information with little privacy concern; whereas user and transaction tends to be identifiable and cart could be sensitive.
- and **they have just a few exceptions**, such as how (a) a full address could be identifiable, but just the state or zip code on their own could be an exception or (b) a user structure could be identifiable but an SSN would be private.

Although I agree that machine reading of privacy metadata is more difficult, I think there is consensus that most programmers (i.e. the site implementer) **will be writing the data, but they wont be reading it.** Those few people or applications that will be reading the data are likely to have the time to write some reusable libraries that facilitate for parsing the information. (I.e. if your job needs it, parsing the rules will be the least of your challenges.



{"default" : ["sitename.com", "ajax.googleapis.com", "ensighten.com"], "analytics" : ["google-analytics.com", "omniture.com"], "personalisation" : ["baynote.com, criteo.com "], "recommendations" : ["doubleclick.net"] }

Annotated Privacy Approach

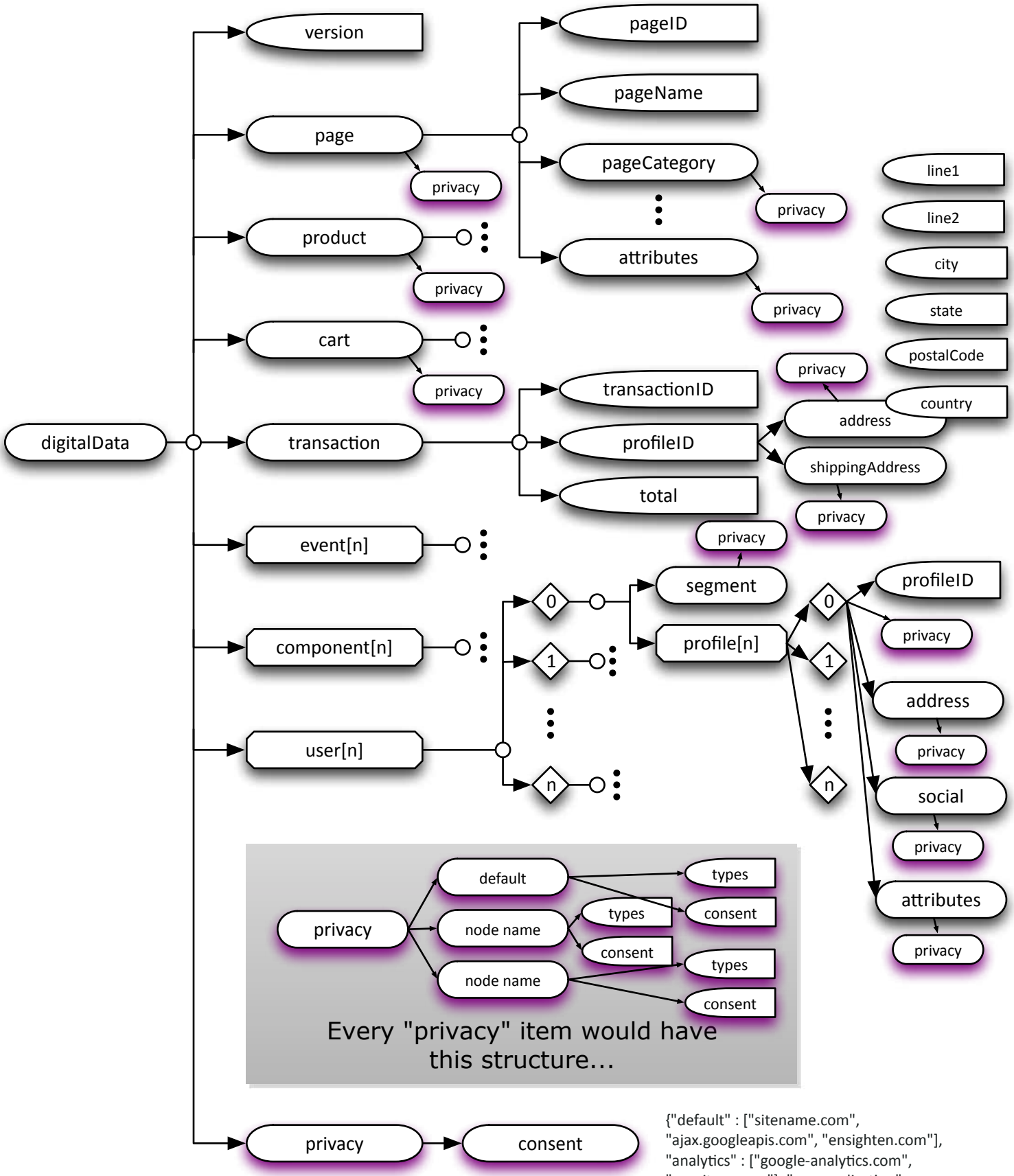
Properties of the Annotated Approach		
Impact on core data	Severe	In certain areas, the amount of privacy data could be equal to the core data itself.
Verbosity	Large	Every item with privacy information must explicitly state all privacy categories and all consent groups.
Complexity	Moderate	Although the inner structure of the privacy object is straightforward, placing the correct objects (whether .privacy object annotations or .*-privacy node annotations) in the right places could make the digitalData structure unwieldy.
Writing Privacy / Implementation	Difficult	The system that is building the digitalData structure must remember the explicit privacy type / consent category list for every portion of the digitalData structure and must remember to write/add each privacy or *-privacy object through the data tree.
Reading Privacy / Implementation	Easy	In principle, every bit of digitalData has straightforward privacy information nearby. The *-privacy sibling nodes could be a little wonky, though.
Enforceability	None	By inter-mingling privacy data with the core digitalData structure, all information is equally accessible to all systems. Privacy becomes an exercise of documentation and self-enforcement only.

Commentary:

It is my firm belief that this solution looks simple on the surface, but will require a huge effort on the part of page developers because of the sheer weight of the privacy “tags” across the structure. In fact, the page developers will probably have to store all of this information in some sort of database any in order for the web server to reconstruct and inject off of the pieces.

It seems that this system has been built in order to make it as easy as possible for a computer programmer to parse and read-in the information (fewer lines of code) but the basic computer programmer is not likely to be needing to read the information—that is the task of analytics and marketing providers who are perfectly well-equipped to write libraries that automate the reading and filtering activity.

In other words, we’re keeping the system simple for those people who are well-equipped to handle some extra complexity. Conversely, we’re making the job of writing the information harder for those people (site builders) for whom we think adoption will be low unless it’s made easy.



{"default" : ["sitename.com", "ajax.googleapis.com", "ensighten.com"], "analytics" : ["google-analytics.com", "omniture.com"], "personalisation" : ["baynote.com, criteo.com "], "recommendations" : ["doubleclick.net"] }

Hybrid Privacy Approach

Properties of the Hybrid Approach		
Impact on core data	Less/Moderate	Some centralization allows ".privacy" to act more as just a protected keyword.
Verbosity	Moderate	While an improvement over the annotated mode, there is still a large amount of nodes across the tree, and each privacy object must list the individual data values of the tree and nest the two privacy lists for each object. (Bulk is the same, but's tucked away better.)
Complexity	Low/Medium	An improvement over the annotated mode (fewer structural exceptions) but the structure within each privacy object must be a bit more complex.
Writing Privacy / Implementation	Difficult	Same argument as for annotated mode. The page developer must place lots of elements all over the place and/or store some comprehensive policy in a database somewhere.
Reading Privacy / Implementation	Easy	Wherever you are, it's easy to check what the policy is for that item.
Enforceability	None	Same as for annotated mode.

Commentary:

If the "Centralized" system is not chosen, this is far cleaner than the extreme annotated method. There are almost no additional complexities (a bit of parsing the internal key map in the branch's privacy object to find the policy that applies to one of the end-nodes) and the clutter should be greatly improved.