Privacy by Design Documentation for Software Engineers Version 1.0

Working Draft 04

Updated 4 June 2014

Technical Committee:

[OASIS Privacy by Design Documentation for Software Engineers (PbD-SE) TC](http://www.oasis-open.org/committees/pbd-se/)

Chairs:

Ann Cavoukian ([commissioner.ipc@ipc.on.ca](mailto:commissioner.ipc@ipc.on.ca)), [Office of the Information & Privacy Commissioner of Ontario](http://ipc.on.ca/), Canada

Dawn Jutla ([dawn.jutla@gmail.com](mailto:dawn.jutla@gmail.com)), [Saint Mary’s University](http://www.smu.ca/)

Editors:

Editor Name(s):

Ann Cavoukian, Office of the Information and Privacy Commissioner of Ontario, Canada, fred.carter@ipc.on.ca

Fred Carter, Office of the Information and Privacy Commissioner of Ontario, Canada, fred.carter@ipc.on.ca

Dawn Jutla, Saint Mary’s University, [dawn.jutla@gmail.com](mailto:dawn.jutla@smu.ca)

John Sabo, Individual, john.annapolis@verizon.net

Frank Dawson, Nokia, frank.dawson@nokia.com

Jonathan Fox, Intel Corporation, [Jonathan\_Fox@McAfee.com](mailto:Jonathan_Fox@McAfee.com)Tom Finneran, Individual, tfinneran@gmail.com

Related work:

This specification is related to:

* Privacy Management Reference Model and Methodology (PMRM) Version 1.0 Committee Specification 01 (July 2013) <http://docs.oasis-open.org/pmrm/PMRM/v1.0/cs01/PMRM-v1.0-cs01.pdf>

Declared XML namespaces:

* <http://docs.oasis-open.org/pbd-se/ns/pbd>
* <http://docs.oasis-open.org/pbd-se/ns/pbdse>

Abstract:

This specification describes a methodology to help engineers to model and document *Privacy by Design* (PbD) requirements, translate the principles to conformance requirements within software engineering tasks, and produce artifacts as evidence of PbD-principle compliance.

Status:

This [Working Draft](http://www.oasis-open.org/committees/process.php#dWorkingDraft) (WD) has been produced by one or more TC Members; it has not yet been voted on by the TC or [approved](http://www.oasis-open.org/committees/process.php#committeeDraft) as a Committee Draft (Committee Specification Draft or a Committee Note Draft). The OASIS document [Approval Process](http://www.oasis-open.org/committees/process.php#standApprovProcess) begins officially with a TC vote to approve a WD as a Committee Draft. A TC may approve a Working Draft, revise it, and re-approve it any number of times as a Committee Draft.

Initial URI pattern:

<http://docs.oasis-open.org/pbd-se/pbd-se/v1.0/csd01/pbd-se-v1.0-csd01.doc>

**Citation format:**

When referencing this specification the following citation format should be used:

**[PbDSE-1.0]** *Privacy by Design Documentation for Software Engineers (PbD-SE) Version 1.0*, 11 June 2014, OASIS Working Draft 01, Revision 4, http://docs.oasis-open-org/…/pbd-se-v1 0-wd04.docx

Copyright © OASIS Open 2014. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](http://www.oasis-open.org/policies-guidelines/ipr) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Table of Contents

1 Introduction 5

1.1 Context and Rationale 5

1.2 Objectives 5

1.3 Intended Audience 6

1.4 Outline of the Specification 6

1.5 Terminology 6

1.6 Normative References 7

1.7 Non-Normative References 7

2 *Privacy by Design* for Software Engineers 9

2.1 Review of the PbD Principles and their Purposes 9

2.1.1 Proactive not Reactive; Preventative not Remedial 10

2.1.1.1 Demonstrable Leadership 10

2.1.1.2 Defined Community of Practice 10

2.1.1.3 Proactive and Iterative 10

2.1.2 Privacy as the Default 10

2.1.2.1 Purpose Specificity 11

2.1.2.2 Limiting Collection, Use, and Retention 11

2.1.2.3 Limiting Collection 11

2.1.2.4 Collecting by Fair and Lawful Means 12

2.1.2.5 Collecting from Third Parties 12

2.1.2.6 Uses and Disclosures 12

2.1.2.7 Retention 12

2.1.2.8 Disposal, Destruction and Redaction 12

2.1.3 Privacy Embedded in Design 13

2.1.3.1 Holistic and Integrative 13

2.1.3.2 Systematic and Auditable 13

2.1.3.3 Reviewed and Assessed 13

2.1.3.4 Human-Proof 13

2.1.4 Full Functionality — Positive-sum, Not Zero-sum 13

2.1.4.1 No Loss of Functionality 14

2.1.4.2 Accommodate Legitimate Objectives 14

2.1.4.3 Practical and Demonstrable Results 14

2.1.5 End to End Security – Lifecycle Protection 14

2.1.5.1 Protect Continuously 14

2.1.5.2 Control Access 14

2.1.5.3 Use Metrics 14

2.1.6 Visibility and Transparency – Keep it Open 14

2.1.6.1 Open Collaboration 14

2.1.6.2 Open to Review 15

2.1.6.3 Open to Emulation 15

2.1.7 Respect for User\* Privacy – Keep it User-Centric 15

2.1.7.1 Anticipate and Inform 15

2.1.7.2 Support Data Subject Input and Direction 15

2.1.7.3 Encourage Direct User/Subject Access 15

3 Operationalizing the PbD Principles in Software Engineering 16

3.1 Organizational Privacy Readiness 16

3.2 Scope and Document Privacy Requirements 16

3.3 Conduct Privacy Risk Analysis and Privacy Property Analysis 17

3.4 Identify privacy resource(s) to support the Solution Development Team 17

3.5 Assign Responsibility for PbD-SE Operationalization and Artifacts Output 17

3.6 Design 18

3.7 Review Code 18

3.8 Plan for Retirement of Software Product/Service/Solution 18

3.9 Review Artifacts throughout the PDLC 18

3.10 Sign off with PbD-SE methodology check list 19

4 Mapping of Privacy by Design Principles to Documentation 21

5 Software Development Life Cycle Documentation for Privacy by Design 25

5.1 *Privacy by Design* Use Case Template for Privacy Requirements 25

5.2 Modeling Representations for Privacy Requirements Analysis & Design 27

5.2.1 Spreadsheet Modeling 27

5.2.2 Modeling Languages 28

5.2.2.1 Privacy by Design and Use Case Diagrams 28

5.2.2.2 Privacy by Design and Misuse Case Diagrams 30

5.2.2.3 *Privacy by Design* and Activity Diagrams 30

5.2.2.4 *Privacy by Design* and Sequence Diagrams 33

5.3 Privacy by Design and Privacy Reference Architecture 33

5.3.1 Privacy Properties 34

5.4 Privacy by Design and Design Patterns 36

5.5 Coding / Development 36

5.6 Testing / Validation 36

5.6.1 *Privacy by Design* Structured Argumentation 36

5.7 Deployment Phase Considerations 36

5.7.1 Fielding 36

5.7.2 Maintenance 36

5.7.3 Retirement 36

5.8 Privacy Checklists 36

Appendix A. Acknowledgements 38

Revision History 39

# Introduction

The OASIS *Privacy by Design* Documentation for Software Engineers Technical Committee provides a specification of a methodology to help engineers model and document *Privacy by Design* (PbD) requirements, translate the PbD principles to conformance requirements within software engineering tasks, and produce artifacts as evidence of PbD-principle compliance. Outputs of the methodology document privacy requirements from software conception to retirement, thereby providing a plan around compliance to *Privacy by Design* principles, and other guidance to privacy best practices, such as NIST’s 800-53 Appendix J [NIST 800-53] and the Fair Information Practice Principles (FIPPs) [PMRM-1.0].

The PbD-SE specification helps engineers to visualize, model, and document PbD requirements and embed the principles within software engineering tasks. Visualization helps software engineers to accelerate their learning and to translate privacy requirements into their software. At the same time privacy governance processes are acquired and/or supported. The PbD-SE specification encourages flexibility of choice of documentation representations for different software engineering methodologies, ranging from waterfall to agile.

The PbD-SE TC references the OASIS Privacy Management Reference Model and Methodology v1.0 (PMRM), as the PMRM represents a comprehensive methodology for undertaking a thorough analysis of application use cases. The PMRM methodology helps establish a structured linkage between privacy policy requirements and controls and the technical services necessary to instantiate them in systems. A PMRM-derived privacy-focused use case template is part of this specification. While remaining agnostic to choice of modeling language, the PbD-SE uses the OMG software modeling standard UML, and other popular representation languages and tools, including and not limited to, data flow diagrams (DFDs) and spreadsheet modeling, to provide concrete examples of documentation. Software engineers, project managers, privacy officers, data stewards, and auditors, among others, may use the PbD-SE methodology for documenting compliance to PbD principles throughout the entire software development life cycle.

## Context and Rationale

The protection of privacy in the context of software engineering requires normative judgments to be made on the part of software engineers. It has become increasingly apparent that software systems need to be complemented by a set of governance norms that reflect broader privacy dimensions. There is a growing demand for provable software privacy claims, systematic methods of privacy due diligence, and greater transparency and accountability in the design and operation of privacy-respecting software systems, in order to promote wider adoption, gain trust and market success, and demonstrate legal and regulatory compliance.

## Objectives

This specification provides guidance and requirements for engineers to document privacy-enhancing objectives and associated control measures throughout the software development life cycle. This documentation is the output of the specification’s methodology but may be supplemented by artifacts produced from auxiliary privacy processes or services, and procedures for internal independent reviewers to conduct reviews of documentation for explicit adherence to *Privacy by Design* (PbD) guidelines. Artifacts include explicit documentation of functional and non-functional privacy requirements. Examples of artifact representations include, and are not limited to, spreadsheet documentation of compliance tasks and processes, those components of use cases, misuse cases, interface design, DFD diagrams, class diagrams, data flow diagrams, scenario diagrams, activity diagrams that clearly show embedding of PbD principles and associated requirements, business model diagrams that show personal data flows across technology platforms, and diagrams of privacy architectures. The documentation specified by this standard may form part of a larger, organization-wide *Privacy by Design* implementation and approach.

## Intended Audience

The intended audience is primarily software engineers tasked with implementing and documenting functional privacy requirements to show compliance to *Privacy by Design* principles. However, as software engineers operate in larger contexts, this specification should also be of interest to their project managers, business managers and executives, privacy policy makers, privacy and security consultants, auditors, regulators, IT systems architects and analysts, and other designers and users of systems that collect, store, process, use, share, transport across borders, exchange, secure, retain or destroy personal data. In larger organizations, where subject matter experts and organizational stakeholders have clear roles in the SDLC, their contributions may be an explicit part of the documentation. In addition, other OASIS TCs and external organizations and standards bodies may find the PbD-SE useful in producing evidence of compliance to *Privacy by Design* principles.

## Outline of the Specification

This specification provides:

* An expression and explanation of the *Privacy by Design* principles in the context of software engineering. In effect, it closes a communications gap among policymakers, business stakeholders, and software engineers.
* A methodology for an organization and its software engineers to produce and reference privacy-embedded documentation to demonstrate compliance to *Privacy by Design* principles.
* A mapping of the *Privacy by Design* principles to engineering-related sub-principles, and to documentation, and thus PbD-SE compliance criteria.
* Privacy considerations for the entire software development life cycle from software conception to software retirement.
* A Privacy Use Case Template that helps software engineers document privacy requirements and integrate them with core functional requirements.
* A *Privacy by Design* Reference Architecture for software engineers to customize to their context, and Privacy Properties that software solutions should exhibit.
* *Privacy by Design* Patterns (future version of spec)
* *Privacy by Design* for Coding and Deployment (future version of spec)
* Software engineering Documentation Checklists

## Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

**Informational Privacy**: "Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others. Viewed in terms of the relation of the individual to social participation, privacy is the voluntary and temporary withdrawal of a person from the general society through physical or psychological means, either in a state of solitude or small-group intimacy or, when among larger groups, in a condition of anonymity or reserve.", see page 7 of Westin, A, *Privacy and Freedom*, 1967) Information Privacy, then is the discipline of applying privacy principles to digital technology, such as the product of software development.

**Personally Identifiable Information (PII)**: any information about an individual including (1) any information that can be used to distinguish or trace an individual‘s identity, and (2) any other information that is linked or linkable to an individual. Adapted from NIST, Guide to Protecting the Confidentiality of Personally Identifiable Information (PII) Special Publication 800-122 (April 2010)

**Principle:** A fundamental truth or proposition that serves as the foundation for a system of belief or behaviour or for a chain of reasoning (Oxford Dictionary); a comprehensive and fundamental law, doctrine, or assumption (Merriam-Webster)

**Privacy Control**: A process designed to provide reasonable assurance regarding the achievement of stated privacy properties or objectives

**Privacy Service**: A service-based software implementation of one or more privacy controls.

**Software Engineer**: A person that adopts engineering approaches, such as established methodologies, processes, architectures, measurement tools, standards, organization methods, management methods, quality assurance systems and the like, in the development of large scale software, seeking to result in high productivity, low cost, controllable quality, and measurable development schedule (Wang, 2011). Note that we include apps that scale to millions of users as “large scale” software.

**Software Organization**: Any organization or department or unit within an organization that engages in the development of software products and services either directly or indirectly.

## Normative References

[**Cavoukian 2011]**

Seven Foundational Principles, available at <www.privacybydesign.ca/index.php/about-pbd/7-foundational-principles/>

**[PbD-FIPPS]**

Cavoukian, A., *The 7 Foundational Principles: Implementation and Mapping of Fair Information Practices*, <http://bit.ly/vjvrPE> January 2011

**[PMRM-1.0]**

*OASIS Privacy Management Reference Model and Methodology (PMRM*) Version 1.0 Committee Specification 01 (July 2013) <http://docs.oasis-open.org/pmrm/PMRM/v1.0/cs01/PMRM-v1.0-cs01.pdf>

[RFC2119]

Bradner, S., *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>., IETF RFC 2119, March 1997.

## Non-Normative References

**[BIRO 2009]**

The BIRO Project, Cross-border flow of health information: is 'privacy by design' enough? Privacy performance assessment in EUBIROD. Available at <http://www.ncbi.nlm.nih.gov/pubmed/22562711>.

[**Cavoukian 1995]**

Privacy-Enhancing Technologies: The Path to Anonymity, Volume II, available at <http://www.privacybydesign.ca/content/uploads/1995/03/anoni-v2.pdf>

[**Cavoukian 2011]**

*Privacy by Design*: *The 7 Foundational Principles Implementation and Mapping of Fair Information Practices* at [www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf](http://www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf)

[**Cavoukian 2012]**

*Privacy by Design*: Leadership, Methods, and Results, Chapter 8: <http://link.springer.com/chapter/10.1007%2F978-94-007-5170-5_8>, *5th Int. Conference on Computers, Privacy & Data Protection European Data Protection: Coming of Age,* Springer, Brussels, Belgium.

[**CICA 2014]**

CICA/AICPA Privacy Maturity Model, available at http://www.cica.ca/resources-and-member-benefits/privacy-resources-for-firms-and-organizations/docs/item48094.pdf

**[Dennedy et al 2014]**

[Michelle Finneran Dennedy](http://www.apress.com/author/author/view/id/7426), [Jonathan Fox](http://www.apress.com/author/author/view/id/7427), [Thomas Finneran](http://www.apress.com/author/author/view/id/7428) (2014). *The Privacy Engineer’s Manifesto: Getting from Policy to Code to QA and Value*, Apress, Jan. 2014, 400 pages.

**[Jutla and Bodorik 2005]**

Dawn N. Jutla, Peter Bodorik (2005), Sociotechnical Architecture for Online Privacy, *IEEE Security and Privacy*, vol. 3, no. 2, pp. 29-39, March-April 2005, doi:10.1109/MSP.2005.50.

**[Jutla et al 2013]**

Dawn N. Jutla, [Peter Bodorik](http://www.informatik.uni-trier.de/~ley/pers/hd/b/Bodorik:Peter.html), [Sohail Ali](http://www.informatik.uni-trier.de/~ley/pers/hd/a/Ali:Sohail.html) (2013). [Engineering Privacy for Big Data Apps with the Unified Modeling Language](http://www.smu.ca/commonresources/webfiles/Jutla-Extending_Use_Case_Diagrams_with_Privacy_Services.pdf). *IEEE* [*Big Data Congress 2013*](http://www.informatik.uni-trier.de/~ley/db/conf/bigdata/bigdata2013.html#JutlaBA13): 38-45. Santa Clara.

**[Jutla 2014]**

Evolving OASIS *Privacy by Design* Standards, April 9, 2014, available at <http://csrc.nist.gov/news_events/privacy_workshop_040914/nist_pew_2014dawn_jutla.pdf>

**[Jutla 2014a**]

M2M Vehicle Telematics and the OASIS Privacy by Design Documentation for Software Engineers (PbD-SE), *European Identity and Cloud Conference*, Munich, April 15 2014, slide deck available to attendees.

**[Jutla 2014b]** Overview of the OASIS PbD-SE and PMRM emerging privacy standards, *PRIPARE workshop*, Cyber Security and Privacy Forum, Athens, May 22, 2014.

Evolving OASIS Privacy by Design Standards, April 9, 2014, available at <http://csrc.nist.gov/news_events/privacy_workshop_040914/nist_pew_2014dawn_jutla.pdf>

**[Jutla et al 2014]**

Dawn N. Jutla, Ann Cavoukian, John T. Sabo, Michael Willett, Fred Carter, Michelle Chibba, *Operationalizing Privacy by Design Documentation for Software Engineers and Business: From Principles to Software Architecture*, submitted for journal review, April 21, 2014.

[NIST 800-53] [Security and Privacy Controls for Federal Information Systems and Organizations – Appendix J, Revision 4: Privacy Controls Catalog]

**[Shostack 2014]**

Adam Shostack (2014), *Threat Modeling: Designing for Security*, Wiley, Feb 2014.624 pages.

**[Zachmann 1987]**

J. Zachmann. [A framework for information systems architecture.](http://zachman.com/images/ZI_PIcs/ibmsj2603e.pdf) IBM Systems Journal, Vol. 26. No. 3, 1987.

# *Privacy by Design* for Software Engineers

This section describes the default context of *Privacy by Design* and lays out the meaning of its principles in terms specific to software engineers.

The *Privacy by Design* framework was unanimously approved by international privacy and data protection authorities in as an international standard in October 2010.

The *Privacy by Design* framework consists of seven high-level and interrelated principles that extend traditional Fair Information Practice Principles to prescribe the strongest possible level of privacy assurance. A mapping of PbD principles to the FIPPs is provided below.

Table 2.1 : *Privacy by Design* Principles Mapped to Fair Information Practice Principles

|  |  |  |
| --- | --- | --- |
| **PbD Principles** | **Meta-FIPPs** | **Traditional FIPPs** |
| 1. Proactive Not Reaction; Preventative Not Remedial | Leadership & Goal-Setting | --- |
| 2. Privacy as the Default Setting | Data Minimization | Purpose Specification  Collection Limitation  Use, Retention & Disclosure Limitation |
| 3. Privacy Embedded into Design | Verifiable Methods | --- |
| 4. Full Functionality –  Positive-Sum, not Zero-Sum | Quantitative Results | --- |
| 5. End-to-End Security Full Life-Cycle Protection | Safeguards | Safeguards |
| 6. Visibility and Transparency - Keep it Open | Accountability (beyond data subject) | Accountability  Openness  Compliance |
| 7. Respect for User Privacy  – Keep it User-Centric | Individual Participation | Consent  Accuracy  Access  Redress |

Source: *Privacy by Design*: *The 7 Foundational Principles Implementation and Mapping of Fair Information Practices* at [www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf](http://www.ipc.on.ca/images/Resources/pbd-implement-7found-principles.pdf)

As with traditional FIPPs, PbD principles set forth both substantive and procedural privacy requirements, and can be applied universally to information technologies, organizational systems and networked architectures. This specification prescribes the application of PbD principles to software engineering documentation.

## Review of the PbD Principles and their Purposes

The Seven (7) Foundational Principles of *Privacy by Design* are:

1. Proactive not Reactive; Preventative Not Remedial

2. Privacy as the Default Setting

3. Privacy Embedded into Design

4. Full Functionality - Positive-Sum, Not Zero-Sum

5. End-to-End Security - Full Lifecycle Protection

6. Visibility and Transparency - Keep It Open

7. Respect for User Privacy - Keep It User-Centric

This specification enables software organizations to embed privacy into the design and architecture of IT systems, without diminishing system functionality. The review seeks to aid the whole team and executive level to understand the PbD principles in a software engineering context.

### Proactive not Reactive; Preventative not Remedial

This principle emphasizes early privacy risk mitigation methods, and requires a clear commitment, at the highest levels, to set and enforce high standards of privacy – generally higher than the standards set by laws and regulation. This privacy commitment should be demonstrably shared throughout by user communities and relevant stakeholders in a culture of continuous improvement.

#### Demonstrable Leadership

Software engineering methods and procedures are in place to ensure a clear commitment, at the highest levels, to prescribe and enforce high standards of privacy protection, generally higher than prevailing legal requirements.

#### Defined Community of Practice

Software engineering methods and procedures are in place to ensure that a demonstrable privacy commitment is shared by organization members, user communities and relevant stakeholders.

#### Proactive and Iterative

Software engineering methods and procedures ensure that continuous processes are in place to identify privacy and data protection risks arising from poor designs, practices and outcomes, and to mitigate unintended or negative privacy impacts in proactive and systematic ways.

### Privacy as the Default

This principle emphasizes establishing firm, automatic, limits to all collection, use, retention and disclosure of personal data in a given system. Where the need or use of personal data is not clear, there is to be a presumption of privacy and the precautionary principle is to apply: the default settings are to be the most privacy protective.

This *Privacy by Design* principle:

* has the greatest impact on managing data privacy risks, by effectively eliminating risk at the earliest stages of the data life cycle.
* prescribes the strongest level of data protection and is most closely associated with limiting use(s) of personal data to the intended, primary purpose(s) of collection; and
* is the most under threat in the current era of ubiquitous, granular and exponential data collection, uses, disclosures and retention.

The default starting point for designing all software-enabled information technologies and systems mandates no collection of personally identifying information —unless and until a specific and compelling purpose is defined.

As a rule, default user settings are maximally privacy-enhancing. This rule is sometimes described as “data minimization” or “precautionary” principle, and must be the first line of defense. Non-collection, non-retention and non-use of personal data is integral to, and supports, all of the other PbD principles.

#### Purpose Specificity

Privacy commitments are expressed by documenting clear and concise purpose(s) for collecting, using and disclosing personal information. Purposes may be described in other terms, such as goals, objectives, requirements, or functionalities. In the context of engineering software designs:

* Purposes must be limited and specific; and
* Purposes must be written in such a way so to be amendable to engineering controls.

#### Limiting Collection, Use, and Retention

Software engineering methods and procedures are in place to ensure that personal data is collected, used, disclosed and retained:

* in conformity with the specific, limited purposes;
* in agreement with the consent received from the data subject(s); and
* in compliance with applicable laws and regulations.

Consistent with data minimization principles, strict limits are in place in each phase of the data processing life cycle engaged by the software under development. This includes:

1. Limiting Collection;
2. Collecting by Fair and Lawful Means;
3. Collecting from Third Parties;
4. Uses and Disclosures;
5. Retention; and
6. Disposal, Destruction and Redaction.

#### Limiting Collection

The software engineer ensures techniques, systems and procedures are put in place to:

1. specify essential versus optional personal data to fulfill identified purposes;
2. periodically review information requirements;
3. document explicit individual consent to collect sensitive personal data;
4. monitor the collection of personal data to ensure it is limited to that necessary for the purposes identified, and that all optional data is identified as such;
5. link stated purpose of collection to the data source identification;
6. ensure auditability of legal or business adherence to collection limitation;
7. associate time expirations to collection;
8. establish levels or types of identity such as gradations of non-identifiable, identifiable or identified data collection and processing that need to be supported; and
9. establish limits to collection associated with levels or types of data subject identity.

#### Collecting by Fair and Lawful Means

The software engineer ensures that techniques, systems, and procedures are put in place to

1. review and confirm methods, before they are implemented, that information is obtained   
   (a) fairly, without intimidation or deception, and   
   (b) lawfully, adhering to all relevant rules of law.
2. associate “fair and lawful” collection with the data source(s).

#### Collecting from Third Parties

The software engineer ensures that techniques, systems and procedures are put in place to:

1. ensure that personal data collection from sources other than the individual are reliable ones that also collect information fairly and lawfully. This requires that:
   1. due diligence be performed before establishing a relationship with a third-party data provider.
   2. privacy policies, collection methods, and types of consents of third parties be reviewed before accepting personal data from third-party data sources.
2. document and, where necessary, seek consent where the software develops or acquires additional information about individuals.

#### Uses and Disclosures

The software engineer ensures that techniques, systems and procedures are put in place to:

1. limit all uses and disclosures of personal data to the specified purposes (and for which the individual has provided implicit or explicit consent);
2. differentiate personal data by both type and quantity, and treat accordingly;
3. anticipate emergency and unintended disclosures (e.g. security breaches);
4. assign and observe time expirations associated with uses;
5. tie future uses to the original collection purpose;
6. establish whether selected “secondary” use(s) may be allowed under law;
7. secure individual consent, where necessary, for disclosures to third parties;
8. establish valid justification(s) for all disclosure without subject consent;
9. inform third parties of relevant collection, use, disclosure and retention requirements, and ensure adherence;
10. audit retention limits and resulting destruction; and
11. ensure security of data transfers.

#### Retention

The software engineer ensures that techniques, systems and procedures are put in place to:

1. limit retention no longer than needed to fulfill the purposes (or as required by law or regulations) and thereafter appropriately dispose of such information;
2. document retention policies and disposal procedures;
3. retain, store, and dispose of archived and backup copies of records in accordance with its retention policies;
4. ensure personal data is not kept beyond the standard retention time unless a justified business or legal reason exists for doing so; and
5. consider contractual requirements when establishing retention practices that may be exceptions to normal policies/practices.

#### Disposal, Destruction and Redaction

The software engineer ensures that techniques, systems and procedures are put in place to:

1. regularly and systematically destroy, erase, or make anonymous personal data no longer required to fulfill the identified purposes or as required by laws and regulations;
2. erase or destroy records in accordance with the retention policies, regardless of the method of storage (for example, electronic, optical media, or paper based);
3. dispose of original, archived, backup and ad hoc or personal copies of records in accordance with its destruction policies;
4. carry out disposal in a manner that prevents loss, theft, misuse, or unauthorized access;
5. document the disposal of personal data;
6. within the limits of technology, locate and remove or redact specified personal data about an individual as required; and
7. consider contractual requirements when establishing disposal, destruction, and redaction practices if these may result in exception to the entity’s normal policies.

### Privacy Embedded in Design

This principle emphasizes integrating privacy protections into the methods by which information systems are designed and developed, as well as how the resulting systems operate in practice. A systemic, principled approach to embedding privacy is to be adopted —one that relies upon accepted standards and frameworks. Wherever possible, detailed privacy impact and risk assessments should be carried out, clearly documenting the privacy risks and all measures taken to mitigate those risks, including consideration of alternative design options and the selection of metrics. The privacy impacts of the resulting technology, operation or information architecture, and their uses, should be demonstrably minimized, and not easily degraded through use, misconfiguration or error.

#### Holistic and Integrative

The software engineer ensures that privacy commitments are embedded in holistic and integrative ways through following this specification at a minimum.

#### Systematic and Auditable

The software engineer ensures that a systematic, principled approach is adopted that relies upon accepted standards and process frameworks, and is amenable to external review.

#### Reviewed and Assessed

The software engineer ensures that detailed privacy impact and risk assessments are used as a basis for design decisions.

#### Human-Proof

The software engineer ensures that the privacy risks are demonstrably minimized and not increase through use, misconfiguration, or error.

### Full Functionality — Positive-sum, Not Zero-sum

This principle seeks to accommodate all legitimate interests and objectives in a positive-sum “win-win” manner. When embedding privacy into a given technology, process, or system, it should be done in such a way that functionality is not impaired, and to the greatest extent possible, that all requirements are optimized. All non-privacy interests and objectives must be clearly documented, desired functions articulated, metrics agreed upon and applied, and zero-sum trade-offs rejected as often being unnecessary, in favour of solutions that enable multi-functionality and maximum privacy.

#### No Loss of Functionality

The software engineer ensures that embedding privacy does not impair functionality of a given technology, process or network architecture.

#### Accommodate Legitimate Objectives

The software engineer ensures that all interests and objectives are documented, desired functions articulated, metrics agreed, and trade-offs rejected, when seeking a solution that enables multi-functionality

#### Practical and Demonstrable Results

The software engineer ensures that optimized outcomes are published for others to emulate and to become best practice.

### End to End Security – Lifecycle Protection

This principle emphasizes continuous protection of personal data across the entire domain in question, whether the personal data is at rest, in motion or in use from initial collection through to destruction. There should be no gaps in either protection of, or accountability for personal data. Applied security standards are to assure the confidentiality, integrity and availability of personal data throughout its lifecycle including, among other things, appropriate use of encryption techniques, strong access controls, logging and auditing techniques, and methods of secure destruction.

#### Protect Continuously

The software engineer ensures that personal data is continuously protected across the entire domain and throughout the data life-cycle, from creation to destruction.

#### Control Access

The software engineer ensures that access to personal data is commensurate with its degree of sensitivity, and be consistent with recognized standards and criteria.

#### Use Metrics

The software engineer ensures that applied security standards assure the confidentiality, integrity and availability of personal data and be amenable to verification. The software engineer ensures solutions support user-level and system-level privacy properties and be amenable to verification.

### Visibility and Transparency – Keep it Open

This principle emphasizes the need to establish accountability for software offerings by providing, to relevant stakeholders, appropriate information and timely evidence about whether, and how, the software or system operates according to stated promises and objectives. Typically, the purposes for demonstrating visibility and transparency are to enhance understanding and trust among deployers of the software, provide for informed choices among consumers and other end-users, and to demonstrate compliance to regulatory authorities. Robust visibility and transparency enhance the capacity for independent verification.

#### Open Collaboration

The software engineer ensures that privacy requirements, risks, implementation methods and outcomes are documented throughout the development lifecycle and communicated to project members and stakeholders.

#### Open to Review

The software engineer ensures that the design and operation of software systems demonstrably satisfy the strongest privacy laws, contracts, policies and norms (as required).

#### Open to Emulation

The software engineer ensures that the design and operation of privacy-enhanced information technologies and systems are open to scrutiny, praise and emulation by all.

### Respect for User\* Privacy – Keep it User-Centric

**\* User = Data Subject**

This principle requires architects and operators to keep the interests of the individual user uppermost by offering strong privacy defaults, appropriate notice, and user-centric and user-friendly interfaces. A key objective of this Principle is to empower end-users to play active roles in the management of their own personal data through mechanisms designed to facilitate informed consent, direct access, verification of accuracy, and complaints.

#### Anticipate and Inform

The software engineer ensures that the software is designed with user/data subject privacy interests in mind, and convey privacy attributes (where relevant) in a timely, useful, and effective way.

#### Support Data Subject Input and Direction

The software engineer ensure that technologies, operations and networks allow users/data subjects to express privacy preferences and controls in a persistent and effective way.

#### Encourage Direct User/Subject Access

The software engineer ensures that software systems are designed to provide data subjects direct access to data held about them, and an account of uses and disclosures.

# Operationalizing the PbD Principles in Software Engineering

This section defines a technology governance methodology for operationalizing PbD Principles in Software Engineering tasks. This methodology is agnostic to software development life cycle (SDLC) methodology or organizational structure. It is directed at ensuring the privacy engineering methodology is operationalized so that it is functionally sustained and not solely dependent on the good will and memory of a single individual. Each organization or entity using the methodology will need to right-size steps based on their resources, complexity and size.

The methodology emphasizes an expectation of iteration over one or more of its steps. Several steps address the questions: what’s changed with personal data, and are privacy properties upheld. The methodology documents the processes, policies, standards, guidelines, and so on that are being used to ensure privacy and security requirements are identified and incorporated and addressed in the development process and methodology. It includes pre-existing privacy and/or security risk models.

## Organizational Privacy Readiness

Privacy capabilities maturity varies across size and age of organizations and industry. The output of this methodological step documents the working contexts of software engineers with respect to privacy readiness. Software engineers in medium to large organizations will not produce all the documentation required for this step. Others in their organizations will, but the software engineer must *reference* and show a working knowledge of the content of organizational privacy documentation, including privacy policies and privacy resources, in her/his organization. Because, software engineers in small organizations may take on the responsibility of a CPO/privacy manager/privacy resource in order to comply with PbD principles, the guidance in this step to identify and create privacy-related roles, responsibilities, and accountabilities is paramount.

In order to demonstrate adherence to Privacy by Design principles, an organization SHALL:

* Establish executive leadership and commitment to *Privacy by Design*.
* Identify who in the engineering organization is responsible for *Privacy by Design*. Depending on size and structure of overall organization, this person may hardline or dotted-line report to the CPO or be the CPO.
* Determine these resources’ responsibilities; i.e., is this person responsible for building the organization’s overall privacy program or is limited to the engineering function.
* Identify who are candidates for privacy engineering leads for projects. This person may be both the privacy lead for the organization and for the project, or the tasks may be divided among several people, according to the extent of the firm’s resources.
* Determine privacy resource’s responsibilities across projects; i.e., is the person(s) also the privacy architect and engineer and responsible for QA or does he/she or they lead a team.
* Determine who within the organization is responsible and accountable for privacy within the components of the engineering process and their relationship to the privacy engineering lead for the organization and for the project.
* Determine training, communication and knowledge transfer/management mechanisms to ensure role is functionalized and not personality dependent.

## Scope and Document Privacy Requirements

Based on an analysis of the product (e.g., defined use cases or user stories), the software organization SHALL scope and document initial product privacy requirements. These requirements establish the default conditions for privacy within the product.

Stakeholders SHALL define and document the key user privacy stories, or privacy use cases, using the Privacy Use Template (section 5.1) [RECOMMENDED] or the more comprehensive OASIS PMRM methodology [2013] [RECOMMENDED], and users’ privacy experience requirements that scope the products’ privacy requirements or privacy features.

They SHALL model and classify data and behavior with common tools, e.g..ONE or MORE of spreadsheets, data flow, data models, UML sequence and/or activity diagrams, or equivalent diagrams to scope the integration of privacy user-level functionality and system properties with the software’s functional requirements. These documents may be used in privacy threat analysis.

## Conduct Privacy Risk Analysis and Privacy Property Analysis

For each product/service/solution, the software organization SHALL examine previous risk assessment reports. If they are not up-to-date, the organization SHALL produce a threat assessment report (including documenting threat models, e.g. [Shostack, 2014], privacy impact assessment, and business impact summary.

For each product/service/solution, Step 2 (Section 3.2) produces a privacy requirements report following the Privacy Use Case Template found in section 5 of this specification document

For each product/service/solution, the software organization SHALL produce a privacy controls’ evaluation and selection report. This report shall address how well the selected controls satisfy privacy properties (see Section 5.3.1). The evidence from this and previous steps are used to determine the level of privacy resourcing in the next step.

[Note: Services to automate privacy impact assessments are available.]

The evidence from this and previous steps are used to determine the level of privacy resourcing in the next step.

## Identify Privacy Resource(s) to support the Solution Development Team

Training and awareness programs on privacy fundamentals, the identity of the team’s privacy and security “champion” that liaises with the responsible Privacy Officer within the organization (if one exists), and the product’s privacy vision to set a goal for the privacy impact threshold for the product, SHALL be documented in this step. There is an expectation of ongoing interaction between privacy officer/privacy engineer and software engineers for writing documentation that can be reviewed correctly. Note that a software engineer can take on accountable and responsible roles for privacy in a small startup that is strapped for resources.

Responsibilities of the privacy resource(s) include maintaining PbD Principles in work products/services. Privacy resource(s) would work with the data stewards and other parts of the team in determining the privacy impact of each data attribute Stakeholders, including software engineers, work together to certify that deployed solutions comply with PbD principles.

[Context: Note that the state of the art shows that some software engineering teams are better resourced than others, regardless of levels of privacy threats, and that privacy resources are scarce as software organizations hold costs down. Thus this step may lead to different privacy resource allocation across teams.]

## Assign Responsibility for PbD-SE Operationalization and Artifacts Output

This step documents the engineers’ working environment with respect to privacy engineering readiness. The software organization SHALL:

1. Document who in the engineering organization is responsible for privacy engineering, the engineer’s knowledge of who within the larger organization is responsible for privacy and privacy engineering, and the executive champion for privacy engineering
2. Document the team’s privacy resource’s responsibilities.
3. Document who is the privacy engineering lead for a specific project, and her/his responsibilities
4. Document the engineers responsible for privacy within the components of the engineering process.
5. Document who within the organization the privacy engineering lead or designate works with to address requirements for overall organizational readiness for deployment or release of privacy engineered solution

Organizations may use a framework such as the RACI model --- responsibility, accountability, consulting/collaborative, informed – to document the assignment of various resources for operational PbD-SE and artifact production. This step achieves sharing of the responsibility for PbD principles across the solutions engineering team in a larger project management process. Responsibilities and associated metrics will be tracked throughout the work stream. The output of this step is the documentation of the accountable and responsible resources, as well as those resources that act in a collaborative/consulting manner, and those who simply stay informed.

## Design

The design of software architecture for most systems is usually described “informally and diagrammatically by means of boxes and lines,” (Abowd et al, 1995). A privacy reference architecture SHALL be created and documented as a basis for later software engineering of complementary software classes. Software engineers may use the privacy services-based (SOA) reference architecture, shown in Fig. 5.5 as a high-level guide. This services-oriented architecture (SOA) architecture will be complemented with detailed and contextual architectural viewpoints of the eventual software product/solution consisting of privacy components, connectors, data repositories, and metadata. Design classes, mappings to implementation classes, UI architecture and designs, and selection of technologies, SHALL also be documented output of this stage.

The resulting architecture SHALL satisfy system-level privacy properties, such as minimizing observability, identifiability and linkability with systems use, traceability and auditability, and accountability. Furthermore, the architecture SHALL satisfy user-level properties, including user comprehension, consciousness, choice, and consent around privacy, and support context, confinement (data minimization and proportionality), and consistency.

## Review Code

Software engineers SHALL execute specific privacy tests, formulated early at the privacy requirements specification stage, to examine the quality of code and for compliance issues. Privacy and security metrics around satisfaction of privacy properties (as per Section 3.6) SHALL guide their evaluation. User testing and/or user studies may also inform the outcome of this step. This step also ensures screening of third party code for privacy violations before incorporation in existing software.

## Plan for Retirement of Software Product/Service/Solution

Product/maintenance teams SHALL create privacy ramp-down guidelines in a retirement plan. If software is consumer facing, organizations communicate to consumers that services are shutting down, and may inform about archiving or disclosure/sharing requirements for databases. If the data controller changes, companies have an obligation to inform users, and if applicable give choice as to deletion, before movement to a new data owner. If an organization uses data processors, or third-party service providers, similar communication to data processors are needed. Retirement plans should contain a quality statement around experience on ramp down. For software and data on hardware, security controls, e.g. NIST directives on hard disk erasure, are documented in the retirement plan. In registration countries, organizations notify Data Processing Authority (DPA) that the DBs are not active anymore, and so on.

## Review Artifacts throughout the PDLC

The context of the methodology is through a data maintenance life cycle for a software-engineered product/service. Documents SHALL be completed but also reviewed periodically during this life cycle by different stakeholders. For example, the privacy legal team reviews the privacy document artifacts to ensure compliance to PbD principles.

## Sign off with PbD-SE methodology check list

This step verifies that proper documentation exists. A checklist MAY be useful for managers and responsible stakeholders to assess, at a glance, whether PbD principles are considered and privacy documentation generated and/or referenced.

Table 3.1 shows an example RACI chart that can act as a checklist. It is expected that organizations will have different RACI assignments according to their specific contexts (e.g. size and organization).

Table 3.1 RACI Chart for Software Engineers

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PbD-SE Methodology  Step | Documents to be referenced/produced | Software  Engineer | | Privacy  Resource | Project  Mgmt. | Mgmt. | | Third  Party | User | | Check-list item |
| 3.1 Reference Organization-al Readiness | Privacy Policy  Document | | CI | RACI | CI | ACI | I | | | CI | ✓ |
|  | Privacy Roles/Training Program in Organization | | I | RACI | CI | AI | I | | | I | ✓ |
| 3.2 Scope Privacy Requirements & Reference Architecture | Functional Privacy Requirements, preliminary controls identification & hooks to Reference Architecture | | RA | RACI | ACI | AI | RAI | | | CI | ✓ |
| 3.3 Conduct Risk and Privacy Property Analysis | Traceability diagrams  and other documentation to show consideration of privacy properties | | CI | RACI | CI | AC | CI | | | - | ✓ |
|  | Risk analysis  Final controls identification | | CI | RACI | CI | ACI | CI | | | - | ✓ |
| 3.4 Identify Privacy Resource Allocation | Privacy resource allocation to SE team | | I | RACI | RI | AI | I | | | - | ✓ |
| 3.5 Create  RACI for Producing Artifacts | RACI assignment to artifact production | | RCI | CI | RACI | AI | - | | | - | ✓ |
| 3.6 Customize Privacy Architecture | Privacy Architecture (incl. services identification) | | RA | ACI | A CI | AI | I | | | - | ✓ |
| 3.7 Conduct Periodic Review | Review of Artifacts throughout the PDLC | | RA | CI | RACI | AI | - | | | - | ✓ |
| 3.8 Execute Code Testing & Privacy Evaluation | Testing and evaluation for satisfying privacy properties | | RA | RCI | RA CI | AI | - | | | C | ✓ |
| 3.9 Create Retirement Plan | Plan for retirement of software solution | | CI | RACI | RACI | ACI | I | | | I | ✓ |
| 3.10Sign-off | Sign off with checklist | | RACI | RACI | RACI | AC | - | | | - | ✓ |

# Mapping of *Privacy by Design* Principles to Documentation

Table 4.1 provides a mapping between the seven PbD principles and the documentation that software engineers must or should produce or reference throughout the software development lifecycle - from software conception to retirement. Please note spreadsheets, modeling languages, and other tools or representations may be used on their own or in combination for documentation, as long as they are sufficiently powerful to capture the essence of the software engineering translation of the PbD principles as provided in Table 4.1.

Table 4.1. Mapping of *Privacy by Design* Principles to Software Engineering Referenced and Generated Documentation

|  |  |  |
| --- | --- | --- |
| **PbD Principle** | **PbD Sub-Principle** | **Documentation** |
| 1. Proactive not Reactive; Preventative not Remedial | **1.1–Demonstrable Leadership**: A clear commitment, at the highest levels, to prescribe and enforce high standards of privacy protection, generally higher than prevailing legal requirements.  **1.2–Defined Community of Practice**: Demonstrable privacy commitment shared by organization members, user communities and relevant stakeholders.  **1.3–Proactive and iterative**: Continuous processes to identify privacy and data protection risks arising from poor designs, practices and outcomes, and to mitigate unintended or negative impacts in proactive and systematic ways. | **SHALL** normatively reference the PbD-SE specification  **SHALL** reference assignment of responsibility and accountability for privacy in the organization, and privacy training program.  **SHALL** include assignment of privacy resources to the software project, recording who are responsible, accountable, consulted, or informed for various privacy-related tasks  **SHALL** reference all external sources of privacy requirements, including policies, principles, and regulations.  **SHALL** include privacy requirements specific to the service/product being engineered, and anticipated deployment environments  **SHALL** include privacy risk/threat model(s) including analysis and risk identification, risk prioritization, and controls clearly mapped to risks |
| 2. Privacy by Default | **2.1–Purpose Specificity:** Purposes must be specific and limited, and be amenable to engineering controls  **2.2–Adherence to Purposes:** methods must be in place to ensure that personal data is collected, used and disclosed:  in conformity with specific, limited purposes;  in agreement with data subject consent; and  in compliance with applicable laws and regulations  **2.3–Engineering Controls:** Strict limits should be placed on each phase of data processing lifecycle engaged by the software under development, including:  Limiting Collection;  Collecting by Fair and Lawful Means;  Collecting from Third Parties;  Limiting Uses and Disclosures;  Limiting Retention;  Disposal, Destruction; and Redaction | **SHALL** list all [categories of] data subjects as a stakeholder  **SHALL** document expressive models of detailed data flows, processes, and behaviors for use cases or user stories associated with internal software project and all data/process interaction with external platforms, systems, APIs, and/or imported code. (Examples of expressive models are roughly *equivalent* to UML models)  **SHALL** describe selection of privacy controls and privacy services/APIs and where they apply to privacy functional requirements and risks.  **SHALL** include software retirement plan from a privacy viewpoint |
| 3. Privacy Embedded into Design | **3.1–Holistic and Integrative**: Privacy commitments must be embedded in holistic and integrative ways.  **3.2–Systematic and Auditable:** A systematic approach should be adopted that relies upon accepted standards and process frameworks, and is amenable to external review.  **3.3–Review and Assess:** Detailed privacy impact and risk assessments should be used as a basis for design decisions.  **3.4–Human-Proof:** The privacy risks should be demonstrably minimized and not increase through operation, misconfiguration, or error. | The OASIS PMRM Privacy Use Case Template or the more comprehensive OASIS PMRM methodology [2013] are **RECOMMENDED** for identifying and documenting privacy requirements.  **SHALL** contain description of business model showing traceability of personal data flows for any data collected through new software services under development.  **SHALL** include identification of privacy design principles  **SHALL** contain a privacy architecture  **SHALL** describe privacy UI/UX design  **SHALL** define privacy metrics  **SHALL** include human sign-offs/privacy checklists for software engineering artifacts  **SHALL** include privacy review reports *(either in reviewed documents or in separate report)* |
| 4. Full Functionality: Positive Sum, not Zero-Sum | **4.1–No Loss of Functionality:** Embedding privacy adds to the desired functionality of a given technology, process or network architecture.  **4.2-Accommodate Legitimate Objectives**: All interests and objectives must be documented, desired functions articulated, metrics agreed, and trade-offs rejected, when engineering software solutions.  **4.3–Practical and Demonstrable Results**: Optimized outcomes should be published for others to emulate and become best practices. | **SHALL** treat *privacy-as-a-functional requirement (see section XXX),* i.e. functional software requirements and privacy requirements should be considered together, with no loss of functionality.  **SHALL** show tests for meeting privacy objectives, in terms of the operation and effectiveness of implemented privacy controls or services. |
| 5. End-to-End Lifecycle Protection | **5.1–Protect Continuously:** Personal data must be continuously protected across the entire domain and throughout the data life-cycle from creation to destruction.  **5.2–Control Access:** Controls on access to personal data should be commensurate with its degree of sensitivity, and be consistent with recognized standards and practices.  **5.3–Security and Privacy Metrics:** Applied security standards must assure the confidentiality, integrity and availability of personal data and be amenable to verification  Applied privacy standards must assure user/data subject comprehension, choice, consent, consciousness, consistency, confinement (setting limits to collection, use, disclosure, retention, purpose), and context(s) around personal data at a functional level, and minimized identifiability, linkability, and observability; maximized traceability, audibility and accountability at a systems level, and be amenable to verification. | **SHALL** be produced for all stages of the software development lifecycle from referencing applicable principles, policies, and regulations to defining privacy requirements, to design, implementation, maintenance, and retirement.  **SHALL** reference requirements, risk analyses, architectures, design, implementation mechanisms, retirement plan, and sign-offs with respect to privacy and security.  **SHALL** reference security metrics AND privacy properties and metrics designed and/or deployed by the software, or monitoring software, or otherwise in the organization, and across partnering software systems or organizations. |
| 6. Visibility and Transparency | **6.1–Open Collaboration:** Privacy requirements, risks, implementation methods and outcomes should be documented throughout the development lifecycle and communicated to project members and relevant stakeholders.  **6.2–Open to Review:** The design and operation of software systems should demonstrably satisfy the strongest privacy laws, contracts, policies and industry norms (as required).  **6.3–Open to Emulation:** The design and operation of privacy-enhanced information technologies and systems should be open to scrutiny, improvement, praise, and emulation by others. | **SHALL** *reference*the privacy policies and documentation of all other collaborating stakeholders  **SHALL** include description of contextual visibility and transparency mechanisms at the point of contextual interaction with the data subject (user) and other stakeholders for data collection, use, disclosure, and/or elsewhere as applicable  **SHALL** describe any measurements incorporated in the software, or monitoring software, or otherwise to measure the usage and effectiveness of provided privacy options and controls, and to ensure continuous improvement.  **SHALL** describe placement of privacy settings, privacy controls, privacy policy(ies), and accessibility, prominence, clarity, and intended effectiveness. |
| 7. Respect for User Privacy | **7.1–Anticipate and Inform:** Software should be designed with user/data subject privacy interests in mind, and convey privacy attributes (where relevant) in a timely, useful, and effective way.  **7.2–Support Data Subject Input and Direction:** Technologies, operations and networks should allow users/data subjects to express privacy preferences and controls in a persistent and effective way.  **7.3–Encourage Direct User/Subject Access:** Software systems should be designed to provide data subjects direct access to data held about them, and an account of uses and disclosures. | **SHALL** describe user privacy options (including access), controls, user privacy preferences/settings, UI/UX supports, and user-centric privacy model.  **SHALL** describe notice, consent, and other privacy interactions at the EARLIEST possible point in a data transaction exchange with a user/data subject or her/his automated agent(s) or device(s). |

# Software Development Life Cycle Documentation for *Privacy by Design*

Privacy documentation tools and recommendations are provided in this section. They will help software engineers generate privacy requirements, and visualize and embed these requirements through encapsulated privacy services, components, or patterns in their product designs and implementations.

## *Privacy by Design* Use Case Template for Privacy Requirements

This section describes tools and techniques that software engineers employ for a comprehensive understanding of privacy in a software development project and operationalizing *Privacy by Design* into the requirements analysis phase of the software development life cycle. Software engineers show consideration of privacy when they include user privacy stories or privacy use cases in their functional analysis and designs; follow privacy requirements elicitation methodologies, such as, the *Privacy by Design* Use Case Template (elaborated in [PMRM-01]) that expresses privacy requirements as functional requirements; and use pragmatic diagramming and documentation tools to visualize and enact *Privacy by Design*.

Applying *Privacy by Design* to the software engineering discipline requires “operationalizing” PbD principles. Among other things, this operational focus requires the decomposition of abstract PbD principles, FIPPs, privacy policies and privacy related business processes into structured and detailed SDLC process and documentation artifacts associated with a specific application, system, or code set. At times this decomposition process can be extremely complex. Using a standardized template can help to make this complexity manageable by providing a structure for analysis and exposing a comprehensive privacy picture associated with a specific use case.

Because documentation artifacts memorialize analysis and actions carried out by stakeholders, a Privacy Use Case Model (Template) can aid in their production. Additionally, adopting a Template throughout the organization and across organizations has multiple benefits:

* A standardized use case template can reduce the time and cost of operationalizing PbD and improve the quality and reusability of documentation
* It provides all stakeholders associated with the specified software development project within an organization a common picture and a clearer understanding of *all* relevant privacy components of the project
* It can expose gaps where PbD analysis has not been carried out or where implementation has not been initiated or completed
* It is a tool to map privacy policies, requirements and control objectives to technical functionality
* A standardized template also facilitates the re-use of knowledge for new applications and the extension of *Privacy by Design* principles more broadly throughout an organization
* Finally, where code must bridge to external systems and applications, a standardized template will help ensure that *Privacy by Design* principles extend to the protection of personal data transferred across system and organizational boundaries.

To help foster accessibility, ease of use and wide adoption a privacy use case template should have a simple basic structure, while also supporting the in-depth analysis needed to address the complexity of privacy requirements in a software development project. As noted in Section 1, the OASIS Privacy Management Reference Model and Methodology Technical Specification v1.0 (PMRM) supports this use case model. It represents a comprehensive methodology for developing privacy requirements for use cases. It enables the integration of privacy policy mandates and control requirements with the technical services and the underlying functionality necessary to deliver privacy and to ensure effective privacy risk management. The PMRM is therefore valuable as the foundation for a comprehensive, standardized use case template.

A PMRM-based template provides:

* a standards-based format enabling description of a specific Privacy Use Case in which personal data or personally identifiable information is involved in a software development project
* a comprehensive inventory of Privacy Use Case components and the responsible parties that directly affect privacy management and related software development for the Use Case
* a segmentation of Use Case components, or User Stories, in a manner generally consistent with the comprehensive OASIS PMRM v1.0 Committee Specification
* an understanding of the relationship of the privacy responsibilities of software developers in privacy-embedded use case development vis-à-vis other relevant Use Case stakeholders
* insights into *Privacy by Design* requirements throughout the different stages of the privacy life-cycle
* the capability to expose privacy control requirements and their supporting technical services and functionality within a Use Case boundary and linkages to external privacy management services
* the potential for assessing in an organization essential PbD predicates for software development (privacy training, privacy management maturity, etc.)
* significant value as a tool to increase opportunities to achieve *Privacy by Design* in applications by extracting and making visible required privacy properties.

The template does not specify an implementer’s SDLC methodology, development practices or in-house data collection, data analysis or modeling tools.

Privacy Use Case Template Components:

1. **Use Case Title**
2. **Use Case Category**
3. **Use Case Description**
4. **Applications associated with Use Case**

*(Relevant applications and products requiring software development where personal data is communicated, created, processed, stored or deleted)*

1. **Data subjects associated with Use Case**

*(Includes any data subjects associated with any of the applications in the use case)*

1. **PI and PII and the legal, regulatory and /or business policies governing PI and PII in the Use Case**

* *(The PI and PII collected, created, communicated, processed, stored or deleted within privacy domains or systems, applications or products)*
* (*The policies and regulatory requirements governing privacy conformance within use case domains or systems and links to their sources)*

1. **Domains, Domain Owners, and Roles associated with the Use Case –** Definitions:

* ***Domains*** *- both physical areas (such as a customer location or data center location) and logical areas (such as a wide-area network or cloud computing environment) that are subject to the control of a particular domain owner*
* ***Domain Owners*** *- the participants responsible for ensuring that privacy controls and functional services are defined or managed in business processes and technical systems within a given domain*
* ***Roles*** *- the roles and responsibilities assigned to specific participants and systems within a specific privacy domain*

1. **Data Flows and Touch Points Linking Domains or Systems**

* *Touch points - the points of intersection of data flows with privacy domains or systems within privacy domains*
* *Data flows – data exchanges carrying PI and privacy policies among domains in the use case*

1. **Systems supporting the Use Case applications**

*(System - a collection of components organized to accomplish a specific function or set of functions having a relationship to operational privacy management)*

1. **Privacy controls required for developer implementation**

*(Control - a process designed to provide reasonable assurance regarding the achievement of stated objectives*[*Note: to be developed against specific domain, system, or applications as required by internal governance policies, business requirements and regulations]*

1. **Services and Underlying Functionality Necessary to Support Privacy Controls**

* *Service - a collection of related functions and mechanisms that operate for a specified purpose*

The following illustration highlights these eleven template components. Note that the template is not a hierarchical model. It recognizes, as does the PMRM on which it is based, the overlapping roles of stakeholders having PbD responsibilities and roles in software development.

****

Fig. 5.0 Color-coded Grouping of Steps of the Privacy Use Case Template

**Organizational Stakeholders Responsible for Template Development**

Responsibilities for contributing to the development of a Use Case and providing information related to specific template components will vary (particularly within a large organization) as illustrated in the followingexample:



## Modeling Representations for Privacy Requirements Analysis & Design

The output of the *Privacy by Design* Use Template in section 5.1 may be represented in numerous ways. The current state-of-the-art in industry involves one of or a combination of spreadsheet, DFD, and/or UML representation.

### Spreadsheet Modeling

Spreadsheets may be used to document privacy requirements and considerations. Examples of recorded attributes for a software project include:

Description of Personal Data/Data Cluster

Personal Info Category

PII Classification

Source

Collected by

Collection Method

Type of Format

Used By

Purpose of Collection

Transfer to De-Identification

Security Control during Data Transfer

Data Repository Format

Storage or data retention site

Disclosed to

Retention Policy

Deletion Policy

These fields also document several outputs of the *Privacy by Design* Use Case Template. In addition to these fields, spreadsheet tabs can contain DFDs and models as described in the following subsections. Other spreadsheet tabs may contain designer consideration of privacy properties. When there are multiple good ways to design a solution, the option that provides the least exposure of identity, link-ability to other data that can re-identify a user, or observ-ability of private data or identity, and least complexity is preferred.

### Modeling Languages

An advantage of modeling languages is their expressive power. They enable people, who understand the problem, and people, who design and implement the information technology solutions, to communicate detailed understanding of functional requirements and to clearly represent interactions with multiple stakeholders. Diagramming speeds up the requirements gathering and specification phase in software engineering. In addition, capturing the diagrams in formal documentation provide a useful audit trail.

Different modeling languages are used across industries. The software industry uses several with the Unified Modeling Language (UML) being a popular standard. This specification remains agnostic to the software engineers’ choice of modeling language, when used, but provides UML-based examples of how privacy requirements may be visualized and managed in modeling language-based documentation.

#### *Privacy by Design* and Use Case Diagrams

This section illustrates how software engineers can *visualize and communicate privacy requirements, select controls, and represent them* in black-box services using high-level *use case* *diagrams*.

UML Use Case diagrams aid software engineers to easily visualize the embedding of privacy into their designs as they abstract out and group details. Use cases are composed of many smaller use case scenarios and/or user stories. As many systems are too complex to be represented in one page, software engineers utilize larger component use cases to hide the system complexity and to handle scale in the use case diagram. There is a similar scaling problem when representing privacy requirements in use case/user story diagrams.

UML is extensible. It may use <<stereotypes>> to reduce the clutter and support scaling as the number of privacy service operations that are required increases. The software engineer may use a Privacy ServicesContainer or a Container stereotype, for example, as is shown in in Fig. 5.1 [Jutla et al, 2013]. The ServicesContainer will host all the privacy services required for a use case diagram and reduce the diagram’s complexity by avoiding the clutter of multiple instances of a privacy service on a use case diagram.

By using components that are well understood by modelers, and directly hooked to privacy services, the software engineer can document privacy requirements for the use case scenario, or user stories. An example is provided below.

**Example to illustrate the concept of PbD-extended Use Case Diagrams:**

In Fig. 5.1, the “super” privacy ServicesContainer is on the communication line between the scientist and the use case scenario. It contains three privacy services: (i) The first privacy service implements the control that requires showing the privacy notice, on the use of output data by the program, to the scientist and obtaining an agreement from her/him. (ii) The second service implements the pseudonymization control for the data before it is used by an application/app. (iii) The third service specifies that anonymization (or de-identification) control is to be applied. As the connection line in Fig. 5.1 from the data scientist is connected to the privacy container, and since the communication line from the container is connected to the sub use case, all privacy services and the controls they represent within the container apply.

Fig. 5.2 shows a more complex UML use case diagram. It is the same scenario as in the previous case, with the addition of a doctor who needs to review recommended treatments, and two further actors. The doctor also needs to be presented with the privacy notice, and the system also needs the doctor’s agreement to the conditions specified in the notice that may involve conditions from a patient’s consent directive. Data shown to the doctor needs to be pseudonymized. For this case scenario an additional requirement is that the system must communicate with the scientist and also the doctor over secure channels. How these privacy requirements are represented using privacy services implementing controls is shown in Fig. 5.2. As in the previous case the data scientist is connected to the container, signifying that all privacy services apply.

All privacy requirements, specified by privacy services within the container, apply to the scientist and the View alternative treatments use case scenario: communication must be over a secure channel, pseudonymization on input data must apply, privacy notice must be given and agreement obtained, and anonymization must be applied on output data. For the doctor, only three controls apply: communication over the secure channel, pseudonymization of data, and privacy notice and agreement. Anonymization is not applied. Consequently, the doctor actor is connected directly to the applicable privacy controls within the container. Furthermore, the doctor’s communication lines need to be labeled to properly identify the connections between the doctor actor, applicable privacy controls, and the doctor’s sub use case.

Further in Fig. 5.2, the data scientist actor, as before is connected to the privacy container signifying that all privacy controls within the container apply to the data scientist’s interaction with the View alternative patients treatments use case scenario. However, there is additional detail in that there are two anonymization methods specified within the Anomymization control. Suppose now that the data scientist has requested and received full and extended access to an anonymized version of the big data set in order to troubleshoot a problem issue. As the scientist is connected to the container, and not directly to the control, the default method, k-anonymity with large k, is specified for her. The public researcher is a new actor that accesses data on which a strong anonymization method, based on the concept of l-diversity, is applied. Another new actor is a head nurse who views a specific treatment record – only secure communication is required. The nurse actor is connected to the Security control and then to the View treatment sub use case. The doctor is now connected to two sub use cases. The doctor is connected to the Review Recommended Treatments sub use case, which requires pseudonymization, notice and agreement, and secure connection. He/she is also connected to the View treatment sub use case – in which case only a secure connection is required.



**Fig. 5.1** Single-actor use case diagram with privacy services implementing controls [Jutla et al, 2013]



**Fig. 5.2** Visualizing Privacy Requirements with multiple-actor use case diagram with privacy services implementing controls [Jutla et al, 2013]

#### *Privacy by Design* and Misuse Case Diagrams

UML Misuse Cases and Misuse Case Diagrams highlight the ways actors can violate stakeholder privacy. They add a view from threat modeling (See [Shostack, 2014]). [Description and EXAMPLE tabled for a future version; includes Figs. 5.3, 5.4 & 5.5]

#### *Privacy by Design* and Activity Diagrams

UML Activity Diagrams can be used at multiple levels of detail. What we call the Business Activity Diagram can be used for the two highest levels of the Zachman Framework [1987]. Software engineers can develop Enterprise Activity Diagrams to show business function relationships at the highest level of the enterprise.

Getting to the next level, they may use Business Activity Diagrams for each project and for each use case. The Activity Diagram shows process relationships and key decisions, and much more information than Use Case Diagrams. Use Case Diagrams are valuable to show the relationships of actors to the various use cases and privacy controls. But the Business Activity Diagram is more valuable for detailed analysis.

Figure 5.6 illustrates the Hospitality company’s Business Activity Diagram as a process modeling tool. We add the Activity Diagram Object icon to show major data attributes tied to processes and decisions to highlight and document where privacy concerns need to be addressed. Business activity diagrams consider and express privacy details without needing to use UML extensions. Figure 5.7 illustrates the use of Business Activity Diagrams with privacy impacting data attributes.

System Activity Diagrams support the design and documentation of modules within a system design. The System Activity Diagram (Figure 5.8) is augmented with an UML Activity Diagram Note icon to show privacy principles and to show which modules address which privacy principles (see Fig. 5.9).



**Fig. 5.6.** Business Activity Diagram for a Vacation Planning project in a Hospitality Company [Dennedy et al 2014]



**Fig. 5.7.** Visualizing Privacy-Impacting Attributes on a Business Activity Diagram [Dennedy et al 2014]



**Fig. 5.8** Privacy Components [Dennedy et al 2014]



**Fig. 5.9** Mapping *Privacy by Design*, GAPP, of FIPPs Principles to Privacy Components [Source: Dennedy et al 2014]

#### *Privacy by Design* and Sequence Diagrams

Software engineers may also visualize privacy requirements by embedding privacy services among functional requirements in UML sequence diagrams (see Fig. 5.10).



**Fig. 5.10** Visualizing Privacy Services in a UML Sequence Diagram

## *Privacy by Design* and Privacy Reference Architecture

A high-level, full stakeholder-view, privacy reference architecture (Fig. 4.1) is provided for customization by a software organization, i.e. any organization that creates and/or uses software to collect and manage client and other stakeholder data. Functional software (e.g. an online social network) that collects data is shown at the bottom level. Functional privacy services (layer 1) are integrated in the functional software through APIs. These privacy services then provide data to a management or integration service layer. These middle-layer services bridge to organizations’ business-related privacy services on collected data at composite layer-3.

Organizations may adopt and customize variants of a privacy reference architecture, *shrinking* or *growing* it depending on their areas of emphases and privacy maturity level. The eight PMRM services of Security, Agreement, Access, Usage, Validation, Interaction, Certification, and Enforcement form a core architectural pattern, as they are repeatable at touch points, i.e. at the interface of two or more stakeholders, applications, systems, data owners, or domains [Jutla et al, 2014]. Additional privacy services, such as for data minimization, complement them at the user interface layer. Data repositories are not shown in Figure 4, due to its services-focus, but software engineers will drill down, possibly guided by this SOA reference architecture, and incorporate data repositories in further architectural viewpoints.



**Fig. 5.11.** Privacy Reference Architecture [ Jutla 2014, 2014a]

### Privacy Properties

Privacy properties may be divided into user-level and systems-level properties. At the user level, we use the 7Cs (Comprehension, Consciousness, Choice, Consent, Context, Confinement, and Consistency [Jutla and Bodorik, 2005] – see Table 5.1 below) as standardized privacy properties for solutions to meet to show respect for individuals as per the seventh PbD Principle. Note that the joint Canadian Institute of Certified Accountants/American Institute of Chartered Public Accountants effort uses 7 criteria for measuring choice and consent in its privacy maturity model [CICA 2014].

At the systems-level, we use ATOIL – Auditability, Traceability, Observ-ability, Identifi-ability, and Link-ability [Jutla et al, 2014, Jutla 2014b]. The Best Information through Regional Outcomes [BIRO 2009] Health project defines the identify-ability privacy property “as a measure of the degree to which information is personally identifiable”; the link-ability privacy property as “a measure of the degree to which the true name of the data subject is linkable to the collected data element”; and observ-ability privacy property as the “measure of the degree to which link-ability and identify-ability are affected by the use of the system” over various contexts as system-level properties. We adopt these definitions. For privacy, the traceability privacy property is important to understand privacy threats or leaks arising from data flowing among software components and systems; the auditability privacy system property is desirable so that the software engineer and her/his company is assured that the system is implementing and executing privacy solutions correctly.

Table 5.1 User-level privacy properties

|  |  |
| --- | --- |
| Comprehension  (User understanding of how PII is handled) | Users should *understand* how personal identifiable information (PII) is handled, who’s collecting it and for what purpose, and who will process the PII and for what purpose across software platforms. Users are entitled to visibility - to know all parties that can access their PII, how to access/correct their own data, the limits to processing transparency, why the PII data is being requested, when the data will expire (either from a collection or database), and what happens to it after that. This category also includes legal rights around PII, and the implications of a contract when one is formed. |
| Consciousness  (User awareness of what is happening and when) | Users should be *aware* of when data collection occurs, when a contract is being formed between a user and a data collector, when their PII is set to expire, who’s collecting the data, with whom the data will be shared, how to subsequently access the PII, and the purposes for which the data is being collected. |
| Choice  (To opt-in or out, divulge or refuse to share PII) | Users should have *choices* regarding data collection activities in terms of opting in or out, whether or not to provide data, and how to correct their data. |
| Consent  (Informed, explicit, unambiguous) | Users must first consent (meaning informed, explicit, unambiguous agreement) to data collection, use, and storage proposals for any PII. Privacy consent mechanisms should explicitly incorporate mechanisms of comprehension, consciousness, limitations, and choice. |
| Context  (User adjusting preferences as conditions require) | Users should/must be able to *change privacy preferences* according to context. Situational or physical context—such as crowded situations (for example, when at a service desk where several people can listen in on your exchange when you provide a phone number, or when you are in the subway with cameras and audio on wearables around you)—is different from when you perform a buy transaction with Amazon.com or provide information to an app registered with an aggregator that sells to advertisers. Data also has context (such as the sensitivity of data, for example, financial and health data) could dictate different actions on the same PII in different contexts. |
| Confinement  (Data minimization, proportionality, and user-controlled re-use of data) | Users must/should be able to *set/request limits* on who may access their PII, for what purposes, and where and possibly when/how long it may be stored. Setting limits could provide some good opportunities for future negotiation between vendors and users. |
| Consistency  (User predictability of outcome of transactions) | Users should *anticipate with reasonable certainty* what will occur if any action involving their PII is taken. That is, certain actions should be predictable on user access of PII or giving out of PII. |

## *Privacy by Design* and Design Patterns

[This section will be developed in a future version.]

## Coding / Development

This section describes software engineering tools and techniques for operationalizing *Privacy by Design* into the coding / development phase of the software development life cycle.

[Note that the name “coding / development” is used instead of “implementation” in order to prevent confusion with implementation in the sense of end-user deployment. Tabled for a future version]

## Testing / Validation

This section describes software engineering tools and techniques for operationalizing *Privacy by Design* into the testing / validation phase of the software development life cycle.

[This section will be developed in a future version.

Validation and Verification – Develop tests at the same time you develop requirements.

Reference OWASP for Penetration testing/ Code Review including code scanning (security)]

### *Privacy by Design* Structured Argumentation

[This section will be developed in a future version.]

## Deployment Phase Considerations

This section describes privacy issues and methods for operationalizing *Privacy by Design* in the deployment phase of the software development life cycle. It is not intended to produce strict documentation guidance. Rather, it is only meant to offer considerations to be taken into account by software engineers.

[This section will be developed in a future version]

### Fielding

[This section will be developed in a future version.]

### Maintenance

[This section will be developed in a future version.]

### Retirement

See subsection 3.8.

## Privacy Checklists

In addition to using Table 3.1 as a checklist, software engineers should use the third column entries in Table 4.1 as a checklist for the documentation, available to auditors, which should be generated within organizations producing software.

## Conformance

This section outlines the requirements that must be met in order for the various "Conformance Targets" of the specification (discussed above) to be considered PbD-SE conforming. It also discusses the relationships between conformance clauses.

See Table 4.1.

1. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

**Participants:**

Peter Brown, Individual

Kim Cameron, Microsoft

Fred Carter, Office of the Information & Privacy Commissioner of Ontario, Canada

Ann Cavoukian, Office of the Information & Privacy Commissioner of Ontario, Canada

Les Chasen, Neustar

Michelle Chibba, Office of the Information & Privacy Commissioner of Ontario, Canada

Frank Dawson, Nokia

Eli Erlikhman, SecureKey technologies, Inc.

Ken Gan, Ontario Canada Lottery and Gaming Corporation

Sander Fieten, Individual Member

Jonathan Fox, Intel Corporation

Frederick Hirsch, Nokia

Gershon Janssen, Individual Member

Dawn Jutla, Saint Mary’s University

Antonio Kung, Trialog

Kevin MacDonald, Individual

Drummond Reed, XDI.org

John Sabo, Individual

Stuart Shapiro, MITRE Corporation

Aaron Temin, MITRE Corporation

Colin Wallis, New Zealand Government

David Weinkauf, Office of the Information & Privacy Commissioner of Ontario, Canada

Michael Willett, Individual

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
| 01 | 10 July 2013 | Ann Cavoukian  Fred Carter | Initial Draft Outline |
| 02 | 20 Aug 2013 | Ann Cavoukian  Fred Carter | Revisions to document structure, content added to sections 1 and 2 |
| 03 | 07 Oct 2013 | Ann Cavoukian  Fred Carter | Incorporate TC member comments and suggested revisions to v02; modify document structure, add new PbD content. Revisions accepted by Committee 30 October 2013 as basis for next revision of working draft. |
| 04 | 6 Mar 2014  07/08 Mar 2014  10 Mar 2014 | Dawn Jutla | Introductory paragraphs added.  Revised Section 3 as a methodology to produce *Privacy by Design* documentation for software engineers; steps refined and re-ordered  Created and added new Section 4. Created and added new mapping of PbD principles to privacy services. This new section links Section 3 and 5.  Restructuring of Section 5. Insertion of new materials.  Refinement of Sections 3 and 4 with TC at March 7 and 8 meetings.  Addition of further steps in methodology |
| 04 | 7 Jan 2014  8 Mar 2014 | Frank Dawson | Provided Spreadsheet modeling for Section 5.  Contributions to methodology in Section 3. |
| 04 | 21 Mar 2014  19 Apr 2014  10 May 2014  31 May 2014  5 Jun 2014 | Dawn Jutla | Filled in initial text for all 10 methodological steps, and created sample RACI table (Fig. 3.1) for the methodology in Section 3.  Substantially revised and completed mapping of PbD-principles to documentation in Section 4 -Table 4.1. Removed Privacy services mapping from this version. Added sections 5.2, 5.2.2, 5.2.2.1, 5.2.2.2, 5.2.2.4 and Architecture section 5.3 and 5.3.1.  Substantial edits throughout sections 3, 4, & 5.  Filled in section 1.4, and multiple edits throughout sections 1, 3, 4, 5, & 6 of document.  Processed John’s edits and added conformance requirements to Section 3 for discussion. |
| 04 | 24 April 2014 | Ann Cavoukian Fred Carter | Completion of Section 2 with 7 PbD principles and sub-principles. Substituted new descriptions of sub-principles into Table 4.1 |
| 04 | 29 April 2014  3 June 2014 | John Sabo | Added Privacy Use Template in Section 5.1.  Feedback and edits throughout |
| 04 | 9 May 2014 | Jonathan Fox | Added bulleted points in Sections 3.1 and 3.5. |
| 04 | 9 May 2014 | Tom Finneran | Created and added Section 5.2.2.3 |