Privacy by Design Documentation for Software Engineers Version 1.0

Working Draft 04

Updated 4 June 2014

Table of Contents

1 Introduction 4

1.1 Context and Rationale 4

1.2 Objectives 4

1.3 Intended Audience 4

1.4 Outline of the Specification 4

1.5 Terminology 4

1.6 Normative References 4

2 *Privacy by Design* for Software Engineers 5

2.1 Proactive not Reactive; Preventative not Remedial 6

2.1.1 Demonstrable Leadership 6

2.1.2 Defined Community of Practice 6

2.1.3 Proactive and Iterative 6

2.2 Privacy as the Default 6

2.2.1 Purpose Specificity 7

2.2.2 Limiting Collection, Use, and Retention 7

2.2.2.1 Limiting Collection 7

2.2.2.2 Collecting by Fair and Lawful Means 8

2.2.2.3 Collecting from Third Parties 8

2.2.2.4 Uses and Disclosures 8

2.2.2.5 Retention 8

2.2.2.6 Disposal, Destruction and Redaction 9

2.3 Privacy Embedded in Design 9

2.3.1 Holistic and Integrative 9

2.3.2 Systematic and Auditable 9

2.3.3 Reviewed and Assessed 9

2.3.4 Human-Proof 9

2.4 Full Functionality — Positive-sum, Not Zero-sum 10

2.4.1 No Loss of Functionality 10

2.4.2 Accommodate Legitimate Objectives 10

2.4.3 Practical and Demonstrable Results 10

2.5 End to End Security – Lifecycle Protection 10

2.5.1 Protect Continuously 10

2.5.2 Control Access 10

2.5.3 Use Metrics 10

2.6 Visibility and Transparency – Keep it Open 10

2.6.1 Open Collaboration 11

2.6.2 Open to Review 11

2.6.3 Open to Emulation 11

2.7 Respect for User\* Privacy – Keep it User-Centric 11

2.7.1 Anticipate and Inform 11

2.7.2 Support User / Data Subject Input and Direction 11

2.7.3 Encourage Direct User / Data Subject Access 11

3 Operationalizing the PbD Principles in Software Engineering 12

3.1 Organizational Privacy Readiness 12

3.2 Scope and Document Privacy Requirements 12

3.3 Conduct Privacy Risk Analysis and Privacy Property Analysis 12

3.4 Identify Privacy Resource(s) to support the Solution Development Team 12

3.5 Assign Responsibility for PbD-SE Operationalization and Artifacts Output 12

3.6 Design 12

3.7 Review Code 12

3.8 Plan for Retirement of Software Product/Service/Solution 12

3.9 Review Artifacts throughout the PDLC 12

3.10 Sign off with PbD-SE methodology check list 12

4 Mapping of *Privacy by Design* Principles to Documentation 13

5 Software Development Life Cycle Documentation for *Privacy by Design* 17

5.1 *Privacy by Design* Use Case Template for Privacy Requirements 17

5.2 Modeling Representations for Privacy Requirements Analysis & Design 17

5.2.1 Spreadsheet Modeling 17

5.2.2 Modeling Languages 17

5.2.2.1 *Privacy by Design* and Use Case Diagrams 17

5.2.2.2 *Privacy by Design* and Misuse Case Diagrams 17

5.2.2.3 *Privacy by Design* and Activity Diagrams 17

5.2.2.4 *Privacy by Design* and Sequence Diagrams 17

5.3 *Privacy by Design* and Privacy Reference Architecture 17

5.3.1 Privacy Properties 17

5.4 *Privacy by Design* and Design Patterns 17

5.5 Coding / Development 17

5.6 Testing / Validation 17

5.6.1 *Privacy by Design* Structured Argumentation 17

5.7 Deployment Phase Considerations 17

5.7.1 Fielding 17

5.7.2 Maintenance 17

5.7.3 Retirement 17

5.8 Privacy Checklists 17

6 Conformance 18

Appendix A. Acknowledgements 20

Revision History 21

# Introduction

## Context and Rationale

## Objectives

## Intended Audience

## Outline of the Specification

## Terminology

## Normative References

# *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 recognized by international privacy and data protection authorities in October 2010. as an essential component of fundamental privacy protection; Privacy and data protection authorities resolved to encourage the adoption of *PbD* principles as guidance to establishing privacy as an organization’s default mode of operation;

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 Reactiven; Preventative Not Remedial | Leadership & Goal-Setting | --- |
| 2. Privacy as the Default Setting | Data Minimization | Purpose SpecificationCollection LimitationUse, Retention & Disclosure Limitation |
| 3. Privacy Embedded into Design | Systematic Methods | --- |
| 4. Full Functionality – Positive-Sum, not Zero-Sum | Demonstrable Results | --- |
| 5. End-to-End SecurityFull Life-Cycle Protection | Safeguards | Safeguards |
| 6. Visibility and Transparency- Keep it Open | Accountability(beyond data subject) | AccountabilityOpennessCompliance |
| 7. Respect for User Privacy – Keep it User-Centric | Individual Participation | ConsentAccuracyAccess Redress |

*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.

This specification enables software engineers to embed privacy into the design and architecture of software-enabled data systems, in order to minimize data privacy risks 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 organizational 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 relevant stakeholders (internal and external) in a culture of continuous improvement.

### Demonstrable Leadership

Software engineering methods and procedures are in place to ensure a clear commitment, from 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 are in place to ensure 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, preferably 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 choice should 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 data —unless and until a specific and compelling purpose is defined.

As a rule, default system 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 data. 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 as functional requirements.

### Limiting Collection, Use, and Retention

The software should be designed in such a way 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 legal requirements.

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:

#### 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. associate sensitivity levels with personal data collected
3. periodically review data requirements;
4. document individual consent to collect sensitive personal data;
5. 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;
6. link stated purpose of collection to the data source identification;
7. ensure auditability of legal or business adherence to collection limitation;
8. assign time expirations to data at time of collection or creation;
9. 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
10. 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 for relevant methods, before they are implemented, that personal data 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 data 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 obtained by third parties be reviewed before accepting personal data from third-party data sources.
2. document and, where necessary, seek consent where the software produces or acquires additional data about individuals.

NOTE: These requirements are specifically for personal data that is collected through a third party. The general requirements as documented in the above sections also apply.

#### Uses and Disclosures

The software engineer ensures 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 exceptional uses and disclosures;
4. assign and observe time expirations associated with uses;
5. tie future uses of personal data to the original collection purpose(s);
6. document whether selected “secondary” use(s) may be allowed under law;
7. secure individual consent, where necessary, for disclosures to third parties;
8. document justification(s) for all disclosures 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 data;
2. document retention policies and disposal procedures;
3. retain, store, and dispose of archived and backup copies of records in accordance with applicable retention policies;
4. ensure personal data is not kept beyond the standard retention period 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 SHALL ensure techniques, systems and procedures are put in place to:

1. regularly and systematically destroy, erase, or de-identify personal data that is no longer required to fulfill the identified purposes;
2. dispose of original, archived, and backup records in accordance with the retention and destruction policies;
3. carry out disposal in a manner that prevents loss, theft, misuse, or unauthorized access;
4. document the disposal of personal data;
5. within the limits of technology, locate and remove or redact specified personal data about an individual as required; and
6. consider contractual requirements when establishing disposal, destruction, and redaction practices if these may result in exceptions to the normal policies/practices.

## Privacy Embedded in Design

This principle emphasizes integrating privacy protections into the methods by which data systems are designed and developed, as well as how the resulting systems operate in practice. A systematic approach to embedding privacy is to be adopted —one that relies upon accepted standards and frameworks. Privacy impact and risk assessments shall be carried out, documenting the privacy risks and measures taken to mitigate those risks, including consideration of alternative design options and choice of metrics. The privacy impacts of the resulting technology, operation or data architecture, and their uses, shall 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 by adopting as broad a scope as possible when identifying and mitigating privacy risks.

### 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 increased 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 shall 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 wherever possible, 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 in the first instance, when seeking a solution that enables multi-functionality

### Practical and Demonstrable Results

The software engineer ensures that, wherever possible, 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 shall 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 system scope 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 is consistent with recognized standards and criteria.

### Use Metrics

The software engineer ensures that security standards are applied that assure the confidentiality, integrity and availability of personal data, and are amenable to verification. The software engineer ensures that solutions support user/data subject-level and system-level privacy properties and are amenable to verification. The reduction of security risks should be quantified and reported regularly.

## Visibility and Transparency – Keep it Open

The software engineer shall create the foundation for for accountable software by providing, to relevant stakeholders, appropriate information and evidence about how the software or system fulfills stated promises and objectives. Demonstrating visibility and transparency enhance understanding among software users, and provide for informed choices by users/data subjects. 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

The software engineer shall 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 users/data subjects to play active roles in managing personal data through mechanisms designed to facilitate informed consent, direct access and control, and redress.

### Anticipate and Inform

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

### Support User / Data Subject Input and Direction

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

### Encourage Direct User / Data Subject Access

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

# Operationalizing the PbD Principles in Software Engineering

## Organizational Privacy Readiness

## Scope and Document Privacy Requirements

## Conduct Privacy Risk Analysis and Privacy Property Analysis

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

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

## Design

## Review Code

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

## Review Artifacts throughout the PDLC

## Sign off with PbD-SE methodology check list

# 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/data subject 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 **SHOULD** reference assignment of responsibility and accountability for privacy in the organization, and privacy training program.**SHALL** include assignment of 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.  |
| **2. Privacy as the 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 user/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** clearly document the purposes for collection and processing, including retention of personal data. **SHOULD include** 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 the mapping of functional and privacy requirements to software components**SHOULD** 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 in 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. | **HALL** 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 risksThe 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** at least contain a privacy viewpoit as part of architecture documentation**SHOULD** define privacy metrics**SHALL** include human sign-offs/privacy checklists for software engineering artifacts**SHOULD** include privacy review reports *(either in reviewed documents or in separate report)***S** |
| **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 in the first instance, 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 2.4),* i.e. functional software requirements and privacy requirements should be considered together, with no loss of functionality.**SHALL** document rationale for defined requirements**SHOULD** show tests for meeting privacy objectives, in terms of the operation and effectiveness of implemented privacy controls or services. |
| **5. End-to-End Security —Lifecycle Protection** | **5.1–Protect Continuously:** Personal data must be continuously protected across the system scope 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 verificationApplied 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** include 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. |
| **These metrics 6. Visibility and Transparency — Keep It Open** | **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 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 user/data subject 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 — Keep it User-Centric** | **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 User/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, wherever feasible, to provide users/data subjects direct access to data held about them, and an account of uses and disclosures. | **SHALL** describe user privacy options (including access), controls, 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 by Design* Use Case Template for Privacy Requirements

## Modeling Representations for Privacy Requirements Analysis & Design

### Spreadsheet Modeling

### Modeling Languages

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

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

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

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

## *Privacy by Design* and Privacy Reference Architecture

### Privacy Properties

## *Privacy by Design* and Design Patterns

## Coding / Development

## Testing / Validation

### *Privacy by Design* Structured Argumentation

## Deployment Phase Considerations

### Fielding

### Maintenance

### Retirement

## Privacy Checklists

# Conformance

This section summarizes the requirements for meeting the PbD “Conformance Targets” of the specification (discussed above)

|  |
| --- |
| **1. Proactive, Not Reactive:** |
| **Project Documentation:** 1. **SHALL** normatively reference the PbD-SE specification
2. **SHALL** reference assignment of responsibility and accountability for privacy in the organization, and privacy training program.
3. **SHALL** include assignment of privacy resources to the software project, recording who are responsible, accountable, consulted, or informed for various privacy-related tasks
4. **SHALL** reference all external sources of privacy requirements, including policies, principles, and regulations.
5. **SHALL** include privacy requirements specific to the service/product being engineered, and anticipated deployment environments
6. **SHALL** include privacy risk/threat model(s) including analysis and risk identification, risk prioritization, and controls clearly mapped to risks
 |
| **2. Privacy as the Default** |
| **Project Documentation:** 1. **SHALL** list all [categories of] data subjects as a stakeholder
2. **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)
3. **SHALL** describe selection of privacy controls and privacy services/APIs and where they apply to privacy functional requirements and risks.
4. **SHALL** include software retirement plan from a privacy viewpoint
 |
| **3. Privacy Embedded into Design** |
| **Project Documentation:** (The OASIS PMRM Privacy Use Case Template is **RECOMMENDED** for identifying and documenting privacy requirements)1. **SHALL** contain description of business model showing traceability of personal data flows for any data collected through new software services under development.
2. **SHALL** include identification of privacy design principles
3. **SHALL** contain a privacy architecture
4. **SHALL** describe privacy UI/UX design
5. **SHALL** define privacy metrics
6. **SHALL** include human sign-offs/privacy checklists for software engineering artifacts
7. **SHALL** include privacy review reports *(either in reviewed documents or in separate report)*
8. **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.
9. **SHALL** show tests for meeting privacy objectives, in terms of the operation and effectiveness of implemented privacy controls or services
10. **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.
11. **SHALL** reference requirements, risk analyses, architectures, design, implementation mechanisms, retirement plan, and sign-offs with respect to privacy and security.
12. **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
 |
| **4. Full Functionality: Positive Sum, not Zero-Sum** |
| **Project Documentation:**1. **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.
2. **SHALL** show tests for meeting privacy objectives, in terms of the operation and effectiveness of implemented privacy controls or services
 |
| **5. End to End Safeguards: Full Lifecycle Protection** |
| **Project Documentation:**1. **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.
2. **SHALL** reference requirements, risk analyses, architectures, design, implementation mechanisms, retirement plan, and sign-offs with respect to privacy and security.
3. **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: Keep It Open** |
| **Project Documentation:**1. **SHALL** *reference*the privacy policies and documentation of all other collaborating stakeholders
2. **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
3. **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.
4. **SHALL** describe placement of privacy settings, privacy controls, privacy policy(ies), and accessibility, prominence, clarity, and intended effectiveness
 |
| **7. Keep it User-Centric** |
| **Project Documentation:**1. **SHALL** describe user privacy options (including access), controls, user privacy preferences/settings, UI/UX supports, and user-centric privacy model.
2. **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).
 |

1. Acknowledgements

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

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Editor** | **Changes Made** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |