**About Building Blocks**

**Building blocks** (BBs) are software modules that can be deployed and combined in a standardized manner. Each building block is capable of working independently, but they can be combined to do much more.

Building blocks are composable, interoperable software modules that can be used across a variety of use cases. They are standards-based, preferably open-source, and designed for scale.

Each building block exposes a set of services in the form of REST APIs that can be consumed by other building blocks or applications.

Please browse the building blocks that are specified in this release in the menu to the left.

**Architecture and Nonfunctional Requirements**

**2 Introduction**

This document is intended to provide guidance for building block working groups and developers of products that will be integrated into a GovStack implementation. It also provides guidlines for implementers and system integrators who are deploying solutions that leverage the GovStack approach. It provides guidelines and principles that should be considered by all building blocks and cross-cutting requirements that must be considered for any GovStack project.

This will accelerate the collaborative development of best-of-breed digital public goods, enhancing efficiency and transparency across the world - especially in low-resource settings.

**2.1 GovStack and the Building Blocks Approach**

GovStack aims to provide a reference architecture for digital governance software to support sustainable development goals. Rooted in a "Whole-of-Government" approach, the GovStack Framework provides a methodology for leveraging common technology components and infrastructure to more easily create and deploy interoperable digital platforms which can address high-priority use cases across multiple sectors. The guidelines and requirements described in this document provide a framework for the development of digital building blocks oriented toward this goal.

**2.2 Criteria for Building Blocks**

The following provide criteria and definitions for Building Blocks, developed by organizations whose work is focused around achievement of the Sustainable Development Goals (SDGs).

**SDG Digital Investment Framework**

The [SDG Digital Investment Framework](https://dial.global/research/sdg-digital-investment-framework/), developed by the International Telecommunication Union (ITU) and the Digital Impact Alliance (DIAL), has formally defined criteria. Building blocks MUST meet the following criteria:

* Reusable software components
* Licensed as open source, proprietary, or freely available with Open Access to data
* Facilitates one or more generic Workflows
* Applicable to multiple SDG Use Cases across multiple sectors
* Interoperable with other ICT Building Blocks
* Designed for Scalability
* Designed for Extensibility
* Standards Based Conformance or Compliance

**Digital Public Goods Alliance (DPGA)**

Additionally, the Digital Public Goods Alliance has created a [definition of Building Blocks](https://digitalpublicgoods.net/DPI-DPG-BB-Definitions.pdf). In this definition, a building block:

* Refers to software code, platforms, and applications that are interoperable, provide a basic digital service at scale, and can be reused for multiple use cases and contexts.
* Serves as a component of a larger system or stack.
* Can be used to facilitate the delivery of digital public services via functions, which may include registration, scheduling, ID authentication, payment, data administration, and messaging.
* Building blocks can be combined and adapted to be included as part of a stack of technologies to form a country’s Digital Public Infrastructure (DPI).
* Building blocks may be open source or proprietary and therefore are not always DPGs.

"Building blocks can be as simple as a common set of rules or protocols (for example email programs like Simple Mail Transfer Protocol - SMTP), or complex (for example an open-source health information system like the DPG, District Health Information Software - DHIS2)“

Characteristics of building blocks:

* Autonomous: building blocks provide a standalone, reusable service or set of services, they may be composed of many modules/microservices.
* Generic: building blocks are flexible across use cases and sectors.
* Interoperable: building blocks must be able to combine, connect, and interact with other building blocks.
* Iterative evolvability: building blocks can be improved even while being used as part of solutions.

Per the DPGA definition, to be considered a building block, solutions must meet the following technical requirements determined by the GovStack Initiative which includes:

1. Open API, Open API Specifications, Rest API
2. Packaged in a container
3. Include a information mediator where communication flows between services that are not co-located

**2.3 Building Blocks**

Building blocks are software modules that can be deployed and combined in a standardized manner. Each building block is capable of working independently, but they can be combined to do much more:

A diagram of a pyramid

Description automatically generated

Building blocks are composable, interoperable software modules that can be used across a variety of use cases. They are standards-based, open source and designed for scale.

Each Building Block represents, as much as possible, the minimum required functionality (MVP) to do its job. This ensures each Building Block is usable and useful on its own, and easily extensible to support a variety of use cases.

A Building Block is composed of domain-driven microservices, modeled as closely as possible on existing roles and processes. This helps ensure each building block is as useful as possible in the real world.

Building Blocks exchange data using lightweight, human-readable data that can easily be extended where needed. Data models and APIs are described in a lightweight manner that’s human-readable, allowing them to be easily and quickly understood and validated.

A diagram of a diagram of a diagram

Description automatically generated

**2.3.1 Building Blocks and UI/UX**

A building block may also be an application which provides re-usable interfaces:

1. An **admin-only form builder** which facilitates building user interfaces (e.g., select questions to be displayed in a maternal-and-child-health registration process)
2. **User interfaces** (i.e., forms) which can be used in lieu of individual end-user apps building their own forms (e.g. I’m building a *new* maternal and child health application; I’d like to use a registration screen flow that’s been pre-built in the registration building block as part of a larger, *composed* application.)
3. A **public API** which exposes the critical **back-end services** performed by this BB (adding a mom to a database; checking for a mom’s enrollment status in a program) to be used (as a microservice) by existing or new applications with legacy/bespoke needs (e.g., i’ve already got a maternal and child health app that the CHWs are using, and I want to send a webhook to the registration BB after a CHW clicks “submit” on our custom form.)

**2.4 Cross-Building Block Communication**

A building block is only so useful on its own. In practice, building blocks MUST be connected together in a secure, robust, trusted manner that facilitates distributed deployments and communications with existing services.

It is STRONGLY RECOMMENDED that a building block use an information mediator (as described below and in the [Information Mediator Building Block specification](https://govstack.gitbook.io/bb-information-mediation)) for any communications across the internet. An Information Mediator is not required for communication between building blocks which are co-located. In this case, communication may occur using standard API calls.

**2.4.1 Federation and Data Exchange Requirements**

Each building block deployment SHOULD use an Information Mediator to federate and communicate with other data consumers and providers, particularly when communicating between services that are not co-located. This ensures the confidentiality, integrity, and interoperability between data exchange parties. An Information Mediator MUST provide the following capabilities:

* address management
* message routing
* access rights management
* organization-level authentication
* machine-level authentication
* transport-level encryption
* time-stamping
* digital signature of messages
* logging
* error handling
* monitoring and alerting
* service registry and discovery

Refer to the full description of the [Information Mediator Building Block](https://govstack.gitbook.io/bb-information-mediation) for more information.

**2.4.2 Organizational Model**

In order to effectively deploy a software solution using the Information Mediator, several policies and processes will need to be applied. This section briefly describes that organizational processes that must be in place.

First, a central operator will be identified and created. This organization will be responsible for the overall operation of the system, including operations and onboarding new members. Policies and contractual agreements for onboarding need to be created.

Trust services need to be set up internally or procured from third parties, including timestamp and certificate authorities. This provides the necessary infrastructure to support distributed deployments.

Finally, members can be onboarded and provided with access to the Information Mediator and methods to register the services that they provide as well as discover services that are available.

Once agreements are in place, members can deploy new services in a decentralized, distributed manner. Before deploying a new service, the central operator must be notified of any changes to access-rights, including organization and machine-level authentication before it can publish or consume data.

**2.4.3 Technical Architecture**

This section provides an overview of the technical processes and architecture that must be implemented once the organizational model has been created.

1. A Central Operator is responsible for maintaining a registry of members, the security policies for building blocks and other member instances, a list of trusted certification authorities and a list of trusted time-stamping authorities. The member registry and security policies MUST be exposed to the Information Mediator.
2. Certificate authorities are responsible for issuing and revoking certificates used for securing and ensuring the integrity of federated information systems. Certificate authorities MUST support the Online Certificate Status Protocol (OCSP) so that an Information Mediator can check certificate validity.
3. Time-stamping authorities securely facilitate time stamping of messages. Time stamping authorities MUST support batched time stamping.
4. The Service Registry provides a mechanism for building blocks to register the services that they provide and for other building blocks to discover and consume those services. Any services provided or consumed by a Building Block that leverages the Information Mediator architecture MUST use this service registry functionality.

**2.5 Keywords and Definitions**

Within this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [BCP 14](https://tools.ietf.org/html/bcp14) [RFC2119](https://tools.ietf.org/html/rfc2119) [RFC8174](https://tools.ietf.org/html/rfc8174) when, and only when, they appear in all capitals, as shown here.

**2.5.1 Building Block-specific definitions**

The following provides definitions for terms that are used by various building blocks.

**Registration:** Any approval/license/certificate issued by a public entity as a result of a request/declaration made by a user of the public service. The result of a “registration” is usually a number and/or a document (called certificate, license, permit, authorization, registration, clearance, approval, etc.)

**Authentication:** This is the technical process of establishing that the credentials (i.e. username, password, biometric etc.) provided by a party (user, system, other) is valid and that the party can be granted basic access to system resources with default access rights. Note that authorization also needs to be applied for a party to access protected resources.

**Authorization:** This is the technical process of establishing whether or not an authenticated party has rights to access a given protected resource. Access rights can typically be granted or revoked administratively on a read-only and/or read-write and/or execute basis through an administrative provisioning process. Permissions or rights defined for a party typically manifest in an access token that is granted at the time of authentication for the party. Hence the processes of authentication and authorization are intrinsically related.

**Workflow Terminology:** See more comprehensive descriptions of the workflow terminology in the [Workflow and Business Process Automation Building Block specification](https://govstack.gitbook.io/specification/building-blocks/workflow).

* (Workflow) Activity - a single step in a workflow process.
* (Workflow) Process - a workflow process contains one or many activities.
* (Workflow) Instance - an instance of execution for a workflow process.

**3 Considerations**

As with any software implementation, there are constraints and limitations in the GovStack approach that must be addressed. In any country context, there will be deficiencies that present challenges to any technology implementation. In the context of GovStack, the constraints and deficiencies that may be present must be considered at the outset of any project.

Here is a list of potential deficiencies that may be encountered with high-level descriptions:

| **Indicator** | **Description** |
| --- | --- |
| ICT Governance | Poor or non-existent National ICT governance structure that makes decisions and ensures accountability framework to encourage desirable behavior in the use of ICT in the country. However, this may be described in documents but the implementation is suboptimal or not enforced. |
| Government ICT policy or Framework | No strategic policy framework for the acquisition and use of IT for social and economic growth in the country. The policy might be at the development stage and where the policy exists, the policy implementation is lagging or non-existent. |
| ICT infrastructure | The development of IT infrastructure in the country is lagging behind or sub optimal because of poor policies and insufficient investments in the ICT sector. Low coverage of power or the national grid and little penetration of alternative sources of energy especially in the rural. |
| Financial Resources and Investments in ICT | Limited funding for ICT projects and initiatives. ICT intervention may not be prioritized. No institutionalized or routinized support for ICT projects/ interventions by the government. |
| ICT projects/ Initiatives | ICT projects and intervention are implemented in a silo, none standard approaches and most of the ICT interventions are proprietary and high cost ventures from private institutions. No national standard architecture for interoperability/ integration of systems |
| Capacity development and social instruments | Low ICT literacy level among user, None or little research and development done by the national institutions/ academia on the use and scale up ICT in the country. Very few ICT professionals to support large scale ICT projects at national level |
| Connectivity/ Internet access | Lack of or minimum network coverage by GSM and or broadband technologies. Low cellular subscribers per capita and very low internet subscribers per capita. The percentage fibre connectivity in the country is low. A greater percentage of the population do not have computers, laptops or smart phones. |
| Access to information | **N**umber of household with internet connectivity is concentrated in the urban areas as opposed to rural areas. |
| Cost competitiveness | Technologies, which are not always ready-for market, are often more expensive than incumbent technologies, without the necessary supportive infrastructure. Competition from existing technologies, including unsustainable technologies |
| Knowledge and skills | New technologies require specialized knowledge and skills, which are often lacking in host countries where education levels in science, engineering and technology can be low, and emerging areas. ICT specialists is low |
| Social Legitimacy | New technologies treated with suspicion in local communities especially if prior experience of job losses or unintended social consequences |
| Cultural barriers | New technologies are seen as a challenge to cultural traditions and communal activities. Technology can also face barriers such as language, role of women in the society, lack of entrepreneurs or dependencies created by decades of development aid |

Additionally, the Principles for Digital Development are especially relevant when designing for low resource setting. Refer to <https://digitalprinciples.org/> for information on these Principles.

Each building block specification SHOULD specify mitigations for these issues.

**4 Building Block Design Principles**

While the following principles are relevant to many technology deployments, when leveraging the GovStack approach it is important to keep these principles in mind during all phases of design, development, and deployment.

**4.1 Citizen-Centric**

Design of systems should be rooted in the needs of the citizens/users of these platforms. A Citizen-centric technology will include the following attributes:

* User-centered design
* Right to be forgotten: everything must be deletable

The best tools evolve from empathizing, understanding and designing for the needs of end-users. Accordingly, we’ve identified a series of use cases and user journeys here: [GovStack Use Cases](https://govstack.gitbook.io/product-use-cases)

Each use case is composed of a collection of modules, or building blocks. As you can see, a relatively small set of these building blocks can be readily applied to a wide variety of applications in low-resource settings.

**4.2 Open**

Where possible, GovStack advocates for the use of open technology, which can reduce cost and help avoid vendor lock-in. Open technology can be defined as:

* Based on open standards
* Based on Digital Development Principles, see <https://digitalprinciples.org/> and <https://digitalinvestmentprinciples.org/>
* Built on open-source software where possible
* Supports open development, see <https://standard.publiccode.net/>
* Cloud native where possible (Docker/Docker Compose/OCI containers)

**4.3 Sustainable**

Any Building Blocks should be developed in a manner which is sustainable and ensures that the technology will continue to be updated and maintained. Some core considerations for sustainability are:

* Stewardship is critical, see <https://publiccode.net/codebase-stewardship/>
* Continuous funding for maintenance, development and evolution
* Attractive to ICT industry and individual developers in deployment environment (incentives must be aligned)
* Lower cost than commercial solutions due to shared development costs
* Uses microservices-based architecture instead of monolithic.
  + This increases interoperability, development and deployment speed and reliability.
  + From Wikipedia: a variant of the [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture) (SOA) structural style – arranges an application as a collection of [loosely coupled](https://en.wikipedia.org/wiki/Loose_coupling) services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

**4.4 Secure**

With any technology deployment, security is paramount. Detailed security requirements are defined in the [GovStack Security Requirements](https://govstack.gitbook.io/specification/security-requirements). Beyond those standards, Building Blocks should have the following attributes:

* Building Blocks are audited and certified before being made available
* Development processes and standards enforce quality and security
* Different certification levels reflect level of standards-compliance
* Regular security scanning and auditing
* Public ratings and reviews
* Comprehensive logging and exception handling

**4.5 Accessible**

It is vitally important that technology solutions be usable by *all*. Some characteristics of accessible design include:

* Meets users where they are: web, mobile, SMS and/or voice. UI supports accessibility technologies, e.g. screen readers.
* SSO allows for signing in once for multiple services
* Shared ownership of code
* Deployment and development processes and standards are open to contributors
* Community-driven development tools for documentation and support
* Blueprints, templates and documentation

**4.6 Flexible**

GovStack is rooted in the concept that Building Blocks should be re-usable and configurable, such that they can support multiple use cases with minimal effort:

* Building Blocks can be reused in multiple contexts
* Each Building Block is autonomous
* Building Blocks are interoperable, adhering to shared standards
* Building Blocks should be easy to set up
* Standardized configuration and communications protocols should be used to connecting Building Blocks
* Building Blocks can be provided as a service (ICT opportunity)

**4.7 Robust**

Deployments of Building Blocks should follow these principles:

* Any client-facing functionality should operate in low-resource environments:
  + Occasional power
  + Low bandwidth
  + Low-reliability connectivity
* Easily scalable for high availability and reliability
* API-only based decoupling
* Asynchronous communications pattern decoupled through rooms is ideal
* Eventual consistency for data

**5 Cross-Cutting Requirements**

Building blocks are responsible for meeting all cross-cutting requirements or specifying why specific requirements do not apply. Govstack compliance and certification processes will validate these requirements.

**5.1 Follow TM Forum Specification REST API Design Guidelines Part 1 (REQUIRED)**

See: [TM Forum REST API Design Guidelines](https://www.tmforum.org/resources/specification/tmf630-rest-api-design-guidelines-4-2-0/)

Some key principles from these design guidelines are as follows:

* APIs MUST not include Personally Identifiable Information (PII) or session keys in URLs - use POST or other methods for this
* MUST support caching/retries
* Resource identification in requests Individual resources are identified in requests, for example using [URIs](https://en.wikipedia.org/wiki/Uniform_resource_identifier) in RESTful Web services. The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server could send data from its database as [HTML](https://en.wikipedia.org/wiki/HTML), [XML](https://en.wikipedia.org/wiki/XML) or as [JSON](https://en.wikipedia.org/wiki/JSON)—none of which are the server's internal representation.
* Resource manipulation through representations. When a client holds a representation of a resource, including any [metadata](https://en.wikipedia.org/wiki/Metadata) attached, it has enough information to modify or delete the resource's state.
* Self-descriptive messages Each message includes enough information to describe how to process the message. For example, which parser to invoke can be specified by a [media type](https://en.wikipedia.org/wiki/Media_type).

**5.2 Follow TM Forum Specification REST API Design Guidelines Parts 2-7 (RECOMMENDED)**

See: [TM Forum REST API Design Guidelines](https://www.tmforum.org/resources/specification/tmf630-rest-api-design-guidelines-4-2-0/)

**5.3 Communicate with other BBs only via API (REQUIRED)**

Paraphrased from the Amazon API Mandate: <https://api-university.com/blog/the-api-mandate/>

1. All BBs must expose their data and functionality through service interfaces (APIs).
2. Building Blocks communicate with each other through these interfaces.
3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team’s data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
4. It doesn’t matter what technology is used. HTTP, Corba, Pubsub, custom protocols — doesn’t matter.
5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

Building blocks MUST NOT use shared databases, file systems or memory for data exchange with other building blocks.

**5.4 APIs must be Versioned (REQUIRED)**

Use semantic versioning when documenting changes to API definitions. Any breaking changes to APIs MUST use a different endpoint, as shown here: e.g. /api/v1 and /api/v2

See <https://semver.org/>

**5.5 Documentation must be Provided (REQUIRED)**

Documentation on the installation and use of the Building Block MUST be provided. Where possible, this documentation SHOULD be stored alongside code in a repository. Documentation MAY be generated from code where applicable.

**5.6 Provide an OpenAPI specification (REQUIRED)**

Each building block’s service APIs MUST be defined and exposed using a standardized machine-readable language. External APIs are described using the OpenAPI 3.x specification. See the following resources for additional information:

* [Definition of the OpenAPI standard](https://swagger.io/docs/specification/about/)
* [Swagger OpenAPI 3.0 Specification](https://github.com/OAI/OpenAPI-Specification/blob/main/versions/3.0.3.md)

**5.7 Building blocks must be deployable as a container (REQUIRED)**

* Each building block MUST be ready to be deployed as independent container images. Source code and build instructions SHOULD be committed to a public code repository where possible.
* A building block may be composed with Kubernetes or docker compose. All build files must be included alongside the source code.

**5.8 Include all deployment scripts (RECOMMENDED)**

When a building block requires deployment tools such as Kubernetes or Ansible, configuration and deployment scripts should be included in the building block repository. Use of this type of deployment configuration will make individual components of the building block independently scalable and make building blocks less monolithic and more efficient.

**5.9 Comply with GDPR Principles (REQUIRED)**

Building Blocks MUST conform to GDPR principles, including the right to be forgotten account deletion, and privacy requirements to protect the rights of individuals. Note that these requirements may vary by region, and building blocks must conform to regulatory requirements wherever they are deployed.

**5.10 Include Support for Capturing Logging information (REQUIRED)**

Building Blocks MUST have a mechanism for generating logging information. This may be as simple as using STDOUT and capturing through docker logs, or may use other log sinking technologies.

**5.11 Use Web Hooks for Callbacks (REQUIRED)**

When Building Blocks require callback functionality, they must use webhooks and not direct links to functions within the building block.

**5.12 Enforce Transport Security (REQUIRED)**

All Building Blocks MUST support secure HTTPS transport with TLS 1.3 and insecure cyphers disabled.

**5.13 GET and PUT APIs must be Idempotent (REQUIRED)**

GET and PUT APIs (as well as HEAD, OPTIONS, and TRACE) must be idempotent, returning the same value if called multiple times. POST and DELETE APIs will not be idempotent as they change the underlying data. Reference <https://restfulapi.net/idempotent-rest-apis/> for more information.

**5.14 Use Stateless APIs wherever Possible to Enhance Scalability (RECOMMENDED)**

API calls SHOULD be able to be made independently of one another. Each API call should contains all of the data necessary to complete itself successfully.

**5.15 Include Transaction/Trace/Correlation IDs (RECOMMENDED)**

Transactions that cross multiple services SHOULD provide a correlation ID that is passed with every request and response. This allows for easier tracing and tracking of a specific transaction.

**5.16 Include Clearly-Defined Key Rotation Policies (RECOMMENDED)**

Some blocks may require the use of security keys. Those that do must have clearly defined key rotation policies to enhance security

**5.17 Databases should not Include Business Logic (RECOMMENDED)**

Database processing tools like triggers and stored procedures should be avoided.

**5.18 Use only Unicode for Text (REQUIRED)**

**5.19 Use ISO8601/UTC for Timestamps (REQUIRED)**

**5.20 Building Blocks must be Autonomous (REQUIRED)**

Each building block MUST be capable of running independently, without requiring other dependencies such as external data stores or other building blocks.

**5.21 Use Secure Configuration (REQUIRED)**

Configuration MUST be done using secure processes, such as environment variables or a secure secret store.

**5.22 Design for Asynchronous First (RECOMMENDED)**

Designs should support occasional connectivity/low bandwidth, and should allow for asynchronous communication between building blocks. A Publish/Subscribe design pattern can be used to handle changes, allowing loosely-coupled solutions to be assembled without changing existing APIs.

**5.23 Use Standardized Data Formats for Interchange (REQUIRED)**

JSON SHOULD be used for data models/services wherever possible. See <https://www.json.org/json-en.html>. Where JSON exhange is not possible, building blocks must use other standard data formats (such as XML).

**5.24 Use Existing Standards for Data Interchange, Where Available (RECOMMENDED)**

If an existing standard is available, it should be used, e.g. DICOM/Hl7/FHIR for healthcare. TMForum has a large library of standardized APIs and data models that can be used.

Building blocks and building block solutions MUST leverage existing standards, especially those listed in the [Standards section](https://govstack.gitbook.io/specification/v/1.0/~gitbook/pdf#pdf-page-VSPzgTsEDUTauDhrndao) below.

**5.25 Use I/O Sanitization (RECOMMENDED)**

Building blocks SHOULD validate all incoming data to ensure that it conforms with the expected format and type. APIs should also sanitize incoming data, removing any unsafe characters or tokens.

**5.26 Provide a Compliance Test Mock/Example Implementation (OPTIONAL)**

A building block MAY provide a mock testing implementation of API functionality to show example endpoints and data payloads. See <https://github.com/GovStackWorkingGroup/bb-template/tree/main/examples> for additional information.

**5.27 Building blocks should be Localizable (RECOMMENDED)**

Where a building block has a human user interaction, it SHOULD be able to present information to the user in their local language. Building blocks should be designed to support multiple locales.

**5.28 Use NTP Synchronization (RECOMMENDED)**

Where precise timestamps are required, building blocks SHOULD leverage Network Time Protocol (NTP) to synchronize timestamps between servers.

**Other Considerations**

Software development best practices are recommended for all building blocks. The following guidelines should be followed as part of the software development process.

**EOL SHOULD be at Least 5 Years**

No languages, frameworks, or dependencies should be used in a building block where that component has an EOL of less than 5 years.

**Preference for TIOBE Top 25 Languages**

See <https://www.tiobe.com/tiobe-index/>

Where possible, building blocks SHOULD be written using commonly used languages to ensure ongoing maintenance and support are as easy as possible. Building blocks MAY leverage less common languages, such as shell scripting where needed.

**Regular Security and Code Quality Audits SHOULD be Run**

These should be run across the code base and dependencies, e.g. <https://www.sonarqube.org/> and/or <https://snyk.io/> .

**SHOULD Include Unit and Integration Test Coverage**

Building blocks should include tests that provide both unit and integration test coverage

**SHOULD Follow Best Practices for Public Code**

See <https://standard.publiccode.net/> and practices outlined here:

* [Code in the Open](https://standard.publiccode.net/criteria/code-in-the-open.html)
* [Bundle Policy and Source Code](https://standard.publiccode.net/criteria/bundle-policy-and-code.html)
* [Create Reusable and Portable Code](https://standard.publiccode.net/criteria/reusable-and-portable-codebases.html)
* [Welcome Contributions](https://standard.publiccode.net/criteria/open-to-contributions.html)
* [Maintain Version Control](https://standard.publiccode.net/criteria/version-control-and-history.html)
* [Require Review of Contributions](https://standard.publiccode.net/criteria/require-review.html)
* [Document Your Objectives](https://standard.publiccode.net/criteria/document-objectives.html)
* [Document your code](https://standard.publiccode.net/criteria/documenting.html)
* [Use plain English](https://standard.publiccode.net/criteria/understandable-english-first.html)
* [Use open standards](https://standard.publiccode.net/criteria/open-standards.html)
* [Use continuous integration](https://standard.publiccode.net/criteria/continuous-integration.html)
* [Publish with an open license](https://standard.publiccode.net/criteria/open-licenses.html)
* [Use a coherent style](https://standard.publiccode.net/criteria/style.html)
* [Pay attention to codebase maturity](https://standard.publiccode.net/criteria/advertise-maturity.html)

**CONSENT BUILDING BLOCK**

**2 Description**

The Consent Building Block enables services for individuals to approve the use of their Personal Data by defining the principles, functions, and architecture of an information system. For organisations that process Personal Data​,​ it provides the ability to know the ​individual's will and legitimately process such Personal Data. The Consent Building Block is a process-oriented GovStack Building Block facilitating auditable bilateral agreements within a multi-agent environment that integrates with most other Building Blocks.

This specification has used several available and recognised open standards below and legal frameworks (such as the [GDPR](https://gdpr.eu/)) for laying the groundwork for its approach to consent.

* [Kantara Initiative](https://kantarainitiative.org/download/7902/): Consent Specification
* [ISO 29184: 2020](https://www.iso.org/standard/70331.html): Online Privacy Notices and Consent
* [ISO/IEC 29100:2011](https://www.iso.org/standard/45123.html): Privacy Framework
* [ISO/TS 17975:2022](https://www.iso.org/standard/78395.html): Health informatics — Principles and data requirements for consent in the Collection, Use or Disclosure of personal health information
* [ISO/IEC TS 27560:2023](https://www.iso.org/standard/80392.html): Privacy technologies — Consent record information structure

**What Consent Is**

In the GovStack context, consent is understood as a voluntary declaration by an individual to approve the processing of their Personal Data. It is one specific justification for Personal Data processing that is assumed to be required by legal or ethical conditions. It assumes that the person can decide on processing their Personal Data, managed in and by other GovStack Building Blocks, and also that the person is free to withdraw their consent at any time.

Some examples of such consent are:

* allowing a healthcare provider to fetch socio-demographic data from a government-run population registry to provide adequate primary healthcare services.
* allowing a government official to fetch relevant data from other/multiple government-run registries to analyse the eligibility for a social benefit programme.
* allow a government official to send Personal Data to a bank for cash transfers on behalf of the government.

**What Consent Is Not**

The use of consent should be avoided in cases as below, which are not part of this specification:

* When a person is simply informed of the processing of the data by the organisation as part of the service provided under contract or by an authority.
* When consent does not have to be obtained in a situation where the entity does not identify or cannot identify people with reasonable effort.

**Assumptions**

Lays out the pre-conditions needed for anyone to use the Consent Building Block.

1. Data Disclosure Agreements between organisations are already in place. For example, a healthcare organisation has already got the required authorisation to use the citizen data registry.
2. To link a Consent Agreement with the specific Individual, Consent Building Block assumes the authentication and authorization to be handled in a trusted manner outside of it (see below).
3. Within the early scope of the Consent Building Block, the act of delegating is kept outside the scope of the Consent Building Block. It is assumed that the authorisation to act on behalf of someone else is already resolved.
4. It is the organization's (a Data Provider or a Data Consumer) obligation to manage and implement internal policies toward its employees relating to their responsibilities for Personal Data processing integrity, specifying it in the employment contract or by other means.

**Consent Agreement Lifecycle**

The life cycle of a consent agreement starts and ends within the organisation responsible for the information system. The organisation knows the context in which the information system operates and the intended purpose of the service. The rules and regulations to be applied for a given level of assurance define the functional framework for consent management.

Consent Building Block deals with transparency on data usage in a given context. Thus privacy-by-design of the system's actors is often an excellent guiding principle for interpreting international, national, and organisational policies and governance principles to implement the functional consent framework. A tangible outcome from a Data Protection Impact Analysis (DPIA) is a structured approach that can deliver the input for the actual implementation of the Consent Building Block.

Individual consent is captured within the context of digital interaction. This interaction is composite of all the information systems involved, not solely the Consent Building Block. Thus, the legal and ethical boundaries of consent are defined in the entirety of the interaction. In particular, consent, as defined by ISO/TS 17975:2015(E), should be seen as a "set of agreements and constraints" that an informed and knowledgeable [individual] agrees to apply to their data processing. This definition, not based on the purpose of the data usage, can lead to a consent management framework also incorporating authorisations or unrelated constraints of the system. For example, in health information and healthcare service delivery, consent is also the process whereby a set of constraints is agreed upon so that information may be collected and used or disclosed. However, it is also the outcome of the process. As a rule of thumb, limiting unintended secondary usage of data is helpful to separate "consent" for a purpose from "consent" as an agreement to constraints and authorisation imposed by the system's functional requirements.

As a result, the organisation responsible for the information system is the driver of the definition of the functional consent management framework. It is also the function of the organisation to design the workflow for obtaining and processing the consent in a way that is purposeful, but not annoying for individuals or data processors with unnecessary bureaucratic overhead. From this framework, the Consent Building Block achieves its purpose by employing Consent Agreements that contain the following:

* A data policy that could be reused across multiple consent agreements (for example, based on the General Data Protection Regulation or any specific regulation)
* The purpose of consent, processed data attributes
* Signatures

A consent agreement life-cycle has four main phases, as illustrated in the figure below:

A screenshot of a diagram

Description automatically generated

[Diagram Source](https://github.com/GovStackWorkingGroup/bb-consent/blob/23Q4.1/spec/images/consent-lifecycle.png)

**Definition**: In this phase, the organisation (a Data Provider or a Data Consumer) adopts and defines a Data Policy that applies to the industry or sector-specific data usage as a template. While this phase is considered a “black box” to the Consent Building Block, it is an essential reference point for configuration and compatibility checks in all following phases.

**Preparation**: In this phase, the organisation (Data Provider or Data Consumer) that intends to process Personal Data configures the Consent Agreement and relevant rules for its use. An organisation could use Personal Data for third-party data sharing, as an example.

**Capture**: In this phase, the Individual can review the Consent Agreement and, once agreed upon, it is captured in a Consent Record by the organisation and stored for verification. This allows an auditor to check and ensure records are in place to process the individual's Personal Data. In the future, this phase could also encompass delegation and other individual use cases.

**Proof**: In this phase, an organisation (A Data Provider or a Data Consumer) can demonstrate that a valid record exists for performing data processing within itself or with other organisations. This allows for internal usage and for an auditor to verify and ensure records are in place to process the individual's Personal Data.

**Actors**

Consent Building Block enables interaction between three (3) distinct user categories, which in combination create the necessary trust framework for the integrity of Personal Data processing. The actors are defined via distinct human roles to be performed in various consent life-cycle phases:

1. Individual as the subject of Personal Data processing;
2. Administrator of the information system exchanging the Personal Data;
3. Data Processing Auditor maintaining independent oversight of the data processing.

Below is the graphical depiction of the actors and their interactions; a more detailed description of the Consent Building Block capabilities is provided in [Chapter 4 - Key Digital Functionalities](https://govstack.gitbook.io/bb-consent/~gitbook/pdf#pdf-page-812DIrIFR0KZQWqzuJVW).

It is important to realize that while the actors are defined via human roles, the consent-related interactions between such roles can be executed in machine-to-machine workflows performing tasks in the interest of the respective actor.

**Interactions with other Building Blocks**

The overall relationship diagram is shown below.

The table below summarises the key relationships consumed during a consent transaction.

| **Building Block** | **Relationship description** |
| --- | --- |
| Identity Building Block | It is assumed the Consent Building Block has already obtained requisite access tokens. |
| Workflow Building Block | Manages the workflow and rules associated with requiring or not requiring consent to use Personal Data. |
| Scheduler Building Block | Provides an engine for time-based triggers to various events of an automated business process, which might also require consent. |
| Information Mediator Building Block | The information mediator Building Block provides a gateway for exchanging data related to consenting workflows; it also provides logs for auditing purposes. |

**3 Terminology**

Terminology used within this specification.

|  |  |
| --- | --- |
| **Term** | **Description** |
| **Configuration** | technical implementation of all the content and process conditions as defined by the Data Policy for Consent Agreement creation, reading, updating and deletion, as well as for providing all necessary actors with the required operations |
| **Consent Agreement** | is the agreement to be signed by the Individual and the Data Controller as prescribed by Data Policy, based on which the Data Providing System may transmit the data to the Data Consuming System for the purposes described in the Consent Agreement. |
| **Consent Record** | is created when an individual signs a consent agreement. It represents a signed consent agreement. |
| **Consent Reference** | a unique identifier used to locate and verify the validity of the Consent Agreement. |
| **Data Providers** | is a legal entity that stores and provides access to an Individual's data, which requires the Individual's consent for processing (outside of its primary purpose/location). |
| **Data Consumers** | is a legal entity that requires the Individual's data from the Data Providers according to the consent of the Individual. |
| **Data Disclosure Agreements** | a Data Disclosure Agreement (DDA) exists between two organisations where one organisation acts as a Data Provider and the other as a Data Consumer. The DDA captures how data is shared between the two organisations and what role and obligation each party has. |
| **Data Policy** | is a formal description of the purpose, nature and extent of consent-based Personal Data processing, covering the configuration needs by Data Providing System and Data Consuming System and the conditions defined by law. |
| **Data Processing Auditor** | is an entity (a person or an organisation) verifying the legitimacy of Personal Data processing by Data Controllers and Data Processors based on the Data Policies and performed tasks. The entity is not to be confused with a data policy auditor that is independent of the actors involved in the operations of consent management and can engage directly with the Consent BB service operator. |
| **Delegate** | the person giving consent (signing Consent Agreement); on behalf of the Individual, |
| **Individual** | is a person about whom the Personal Data is stored in an information system (a.k.a. “Data Subject”) and who agrees or not with the use of this data outside of its primary purpose/location. |
| **Legal Entity** | is an organisation (public or private) ​that has the rights and obligations to define standards for Personal Data processing. E.g. a public health authority |
| **Personal Data** | is any information that (a) can be used to identify the Individual to whom such information relates, or (b) is or might be directly or indirectly linked to the Individual (ISO(IEC 29100:2011) |
| **Regulations** | are broadly defined as rules followed by any system: could be laws, bylaws, ​norms or architectures that​ regulate a given system. The term and definition is inspired by Lessig’s modalities of regulation: <https://lessig.org/images/resources/1999-Code.pdf> |

**4 Key Digital Functionalities**

Key Digital Functionalities describe the core (required) functions that this Building Block must be able to perform.

The Consent Building Block enables organisations to enforce Data Policies that require signed consent by Individuals for the use of their Personal Data. Its key purpose is to allow individuals to view Consent Agreements and sign or withdraw their consent on what Personal Data is used and accessible to organisations. It also clarifies the Data Policy applied, such as the purpose, retention period, jurisdiction, third-party data sharing, etc.

The Consent Building Block implements the key functionalities described in the [consent management lifecycle](https://govstack.gitbook.io/bb-consent/~gitbook/pdf#pdf-page-BdqWOrXqlZpeJDKgZ37l-consent-agreement-lifecycle). It includes the ability to configure consent agreements by an organisation admin, present consent requests towards individuals, capture consents, enable queries if consent exists, or not, and enable independent audit of consents.

The functionalities are derived from the consent agreement lifecycle and categorised according to the [Actors](https://govstack.gitbook.io/bb-consent/~gitbook/pdf#pdf-page-BdqWOrXqlZpeJDKgZ37l-actors) described above. While the consenting workflows (as described above) are implicitly considered the centrepiece of the Consent Building Block, it is important to realise that the integrity of consent management can only be achieved if robust configuration before and auditing after the Consent Agreement signing and Consent Record verification activities are in place. Hence, the functionalities are described following the logical sequence of the consent agreement lifecycle and they are all equally important components of the Consent Building Block.

The Consent process (creating and signing Consent Agreements and Consent Records) is initially managed in the application provided by the Organisation that is legally required to collect the consent. Since it can be either a Data Consuming organisation or a Data Providing organisation, the Consent Building Block allows both to be able to verify their conformance with the underlying Data Policy, both organisations must be able to access and use the application.

While the Actors generally fall in line with the categories of the functionalities, it is important to realise that “auditing” functions in the narrow sense, verifying if data processing is being (or has been) processed according to the Data Policy requiring a consent, is relevant to various entities involved in the data processing. For this reason, the generic “verification” activity may be executed as part of various workflows satisfying the needs of different actors.

Following is the first core set of key functionalities of the Consent Building Block. For potential future developments of the specification follow the work in progress at [GovStack confluence page](https://govstack-global.atlassian.net/wiki/spaces/GH/pages/183205908/Future+Considerations+Consent).

**4.1 Administrator User Functionalities**

The Administrator (Data Provider or Data Consumer Admin) configures the Consent Building Block on behalf of the organisation. For simplicity, it is foreseen that one organisation involved in the data processing transaction (that is either Data Provider or Data Consumer) takes the responsibility for the configuration of the Data Policy and respective Consent Agreements(s), and so that the organisation’s Administrator maintains the required configurations.

The main Administrator actions expected to perform via Consent Building Block are:

* configuring Data Policies, requesting and signing Consent Agreements with Individuals;
* viewing (reading, exporting) the Consent Agreements and relevant reports;
* event-driven (opt-in or opt-out) subscription to (notifications of) changes in Consent Agreements;
* logging and maintaining an auditable overview of all Personal Data transactions according to Consent Agreements as well as configuration versions.

The table below summarises the key use cases identified for an organisation's Administrator. Organisations can be Data Consumers or Data Providers, i.e. the organisations legally delegated the responsibility for collecting consent for the systems handling Personal Data processing.

|  |  |
| --- | --- |
| **Consent use-cases** | **Link(s) to the UCS** |
| CREATE CONSENT AGREEMENT - Here, an organisation Administrator creates a Consent Agreement based on the Data Policy requirements. | [UC-C-PIC-A-001](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-a-organisation-administration-use-cases-configuration#uc-c-pic-a-001-postpartum-and-infant-care-configuration-create) |
| UPDATE CONSENT AGREEMENT - Here, an organisation Administrator updates the Consent Agreement based on the Data Policy requirements. | [UC-C-PIC-A-002](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-a-organisation-administration-use-cases-configuration#uc-c-pic-a-002-postpartum-and-infant-care-configuration-update) |
| READ CONSENT AGREEMENT - Here, an organisation Administrator reads the Consent Agreement. | [UC-C-PIC-A-003](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-a-organisation-administration-use-cases-configuration#uc-c-pic-a-003-postpartum-and-infant-care-configuration-read) |
| DELETE CONSENT AGREEMENT - a special case of consent agreement update. | [UC-C-PIC-A-004](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-a-organisation-administration-use-cases-configuration#uc-c-pic-a-004-postpartum-and-infant-care-configuration-delete) |
| CONSENT AGREEMENT CHANGE NOTIFICATION - Notifications for Data Providing and Data Consuming Systems, as well as Individuals upon changes to Agreement or Policy configuration. | [UC-C-PIC-A-005](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-a-organisation-administration-use-cases-configuration#uc-c-pic-a-005-postpartum-and-infant-care-configuration-notifications) |

**4.2 Individual User Functionalities**

The capabilities for individuals that Consent Building Block supports are:

* viewing and understanding Data Policies applying to their Personal Data processing;
* agreeing and disagreeing with and toggling between the conditions of Personal Data use as described in the Consent Agreement;
* obtaining copies of their Consent Agreement(s);
* delegating their consent rights (out-of-scope for current technical release).

The scope for the current version of Consent Building Block assumes that the Individual is acting for themselves. Ultimately the Consent Building Block will include in the Consenting Process the capacity to sign a Consent Agreement in the name of another individual - to act as the Delegate, which is used as the criterion for technical implementation. However, the Delegate and the Individual relationship is expected to be maintained outside of the Consent Manager, which assumes that the person signing the Consent Agreement (i.e. Consenter) has been authorised to do so.

The table below summarises the key use cases identified for the Individuals.

|  |  |
| --- | --- |
| **Consent use-cases** | **Link to the UCS** |
| VIEW CONSENT - Here, the Individual views the Consent Agreement and the conditions for Personal Data processing (with adequate clarity for informed understanding). This includes obtaining copies of the consent agreement. | [UC-C-PIC-I-001](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-i-individual-use-cases-services#uc-c-pic-i-001-view-agreements) |
| GIVE CONSENT - Here, the Individual signs a Consent Agreement during a data sharing workflow. Note that this can also happen offline without data sharing in place. | [UC-C-PIC-I-002](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-i-individual-use-cases-services#uc-c-pic-i-002-give-consent-to-fetching-data) |
| WITHDRAW CONSENT - Or update existing consent | [UC-C-PIC-I-003](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-i-individual-use-cases-services#uc-c-pic-i-003-withdraw-or-update-existing-consent) |
| Consent agreement change notification | [UC-C-PIC-I-004](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-i-individual-use-cases-services#uc-c-pic-i-004-consent-agreement-change-notification) |

**4.3 Data Processing Auditor User Functionalities**

**Important note**: In the Consent Building Block, we define the Data Processing Auditor's role (see 1.3 Terminology and 1.5.3 Actor definition) as an organisation's auditor implementing the Consent Building Block. The auditor role will most probably be akin to a Data Protection Officer (DPO), possibly from an external third-party organisation and involve activities outside of the Consent Building Block.

To avoid ambiguity, we use the precise term Data Processing Auditor to stress the specificity of tasks to be performed by and for the Consent Building Block; all other actions not within the Consent Building Block's scope are considered as an external prerequisite and as a “black box” activity. With respect to audit, this role is distinguished from the data policy auditor. The Data Processing Auditor relies on an audit universe defined by the control and risk management of the specific project and context (i.e. outside the Consent Building Block). “Who needs to consent to what” is the outcome of a DPIA (Data Protection Impact Analysis), ensuring that the data policies are compliant with the relevant data protection regulations for the project.

The main actions a Data Processing Auditor (or Data Protection Officer, DPO) is expected to perform via Consent Building Block are:

* auditing tracking the consents (opt-in/opt-out);
* auditing tracking Data Policy changes and configuration conformance with it;
* viewing (reading, exporting) the Consent Agreements and relevant reports in a verifiable form.

For the implementation of a specific use case, it is important to distinguish the Data Processing Auditor, an actor described here, from a data policy auditor, an actor of the risk organisation. It is expected that the two roles are coordinated in the risk management process. Within the Consent Building Block, the Data Processing Auditor performs the tasks that will allow a data policy auditor to confirm that the implemented system complies with existing regulations demanding consent.The table below summarises the key use cases identified for the Data Processing Auditor.

**Also to consider:** “READ CONSENT STATUS” use-case is also used by any workflow (and Actor) that requires verification of the consent status (for example, before executing the data transfer from Data Providing System to the Data Consuming System).

|  |  |
| --- | --- |
| **Consent use-cases** | **Link to the UCS** |
| AUDIT CONSENT - Query the Consents related to individuals or policies (opt-in/opt-opt) | [UC-C-PIC-AT-001](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-at-data-processing-auditor-use-cases-audit#uc-c-pic-at-001-consent-postpartum-and-infant-care-audit-consent) |
| MONITOR POLICY UPDATE- Tracking Data Policy changes and configuration conformance with it; | [UC-C-PIC-AT-002](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-at-data-processing-auditor-use-cases-audit#uc-c-pic-at-002-consent-postpartum-and-infant-care-monitor-policy-update) |
| READ CONSENT STATUS - Viewing (reading, exporting) the Consent Agreements and relevant reports in a verifiable form | [UC-C-PIC-AT-003](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-at-data-processing-auditor-use-cases-audit#uc-c-pic-at-003-consent-postpartum-and-infant-care-read-consent-status) |
| VERIFY CONSENT INTEGRITY - Ability to check the integrity of the signed agreements | [UC-C-PIC-AT-004](https://govstack.gitbook.io/bb-consent/v/consent-1.0/internal-use-case-definitions/uc-c-pic-at-data-processing-auditor-use-cases-audit#uc-c-pic-at-004-consent-postpartum-and-infant-care-verify-consent-integrity) |

**Scenarios: Consent and Data Access**

As described above under [Universal Consenting Workflows](https://govstack.gitbook.io/bb-consent/~gitbook/pdf#pdf-page-plDy0hjQEEVMpjJpTMJk-universal-consent-workflow-pre-registration-uses-centralised-id), there may be an unlimited number of business processes that require consent. The following scenarios are but a few examples illuminating how appropriate access to data can and should be handled when processing or consuming data with the support of Consent Building Block functionalities.

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | **Source Building Block** | **Target Building Blocks** | **Description** |
| 1.1 Querying: Which Consent Agreement is needed for specified data processing/ consumption? | Any | Workflow Building Block | Consent Building Block does have knowledge or state to resolve which Data Consumer or Data Producer requires consent. Everything regarding consent has a precondition that a decision is made and manifested in the Workflow Building Block or any other Data Consumer. |
| 1.2 Data processing/ consuming system stores/fetches data with consent + prompts the user if none exists | Any | Consent Building Block | Given an Agreement ID and a User ID, the Consent Building Block can resolve if consent exists and possibly prompt the user. Workflow Building Block is especially inappropriate here because of User Interface integration and a blocking and sequential call stack. |
| 1.3 Data processing/ consuming system stores/ fetches data with consent, no halts operations without consent | Any | Workflow | Given an Agreement ID and a User ID, the Workflow Building Block can complete an atomic action requiring consent. Operations shall not proceed if consent does not exist. |
| 1.4 Data processing/ consuming system stores/ fetches data with consent, consent is prompted asynchronously | Any | Workflow | The Workflow Building Block may halt operations and asynchronously prompt the user for consent if none exists (or is invalid). After fetching consent, the Workflow Building Block should revert to the targeted data consuming/ processing operation. |
| 1.5 Appropriate access to data that does not require consent | Any | Workflow | Not necessarily related to Consent Building Block. |
| 1.2-1.4 Side effects | Workflow |  | Any attempt to read consent and process/consume data is logged and auditable. |
| 2.1 Inappropriate access: Data processing/consuming system inappropriately stores/fetches data without consent | Any | N/a | Any consent-requiring data access is assumed logged.  Auditing of inappropriate data access is only possible when a log trace exists. |
| 3.1 | Workflow | Consent | Given an Individual, query if active Consent Records exist (for instance, to spot if other external data needs to be kept). |
| 4.1 | Any | Consent | Fundamental individual rights (General Data Protection Regulation/ Data Protection Act/ Right to be Forgotten/etc.) |
| 4.2 | Any | Consent | Fundamental individual rights (General Data Protection Regulation/Data Protection Act/etc.) |

**5 Cross-Cutting Requirements**

This section will highlight important requirements or describe any additional cross-cutting requirements that apply to this Building Block.

**5.1 Requirements**

The cross-cutting requirements described in this section are an extension of the cross-cutting requirements defined in the [architecture specification document](https://govstack.gitbook.io/specification/v/1.0/architecture-and-nonfunctional-requirements). Any implementation MUST adhere to all requirements from [GovStack Security Requirements](https://govstack.gitbook.io/specification/v/1.0/security-requirements).

**5.1.1 Personal data MUST be kept private (REQUIRED)**

Personal data MUST be kept private and never shared with any parties, except where specific authorisation has been granted. The Consent Building Block shall follow the privacy principles as laid out in the GovStack architecture.

**5.1.2 Data Policy Audit Logging (REQUIRED)**

Logs MUST be kept in a database of all created, updated, or deleted records. Logs MUST include timestamps and identify the user and affiliation that performed the transaction.

**5.1.3 Follow GovStack Access control requirements (RECOMMENDED)**

In general, the Consent Building Block shall follow the authentication and authorisation requirements as laid out in the [GovStack architecture](https://govstack.gitbook.io/specification/v/govstack-specification-main/security-requirements/4-security-management#4.2.1.1-authentication-and-authorization). For clarity, Consent Building Block's API endpoints are invoked with a client-supplied API key which must defer to the Identification and Verification Building Block in order to verify the role and/or scope of the API key matches the API endpoint to which it is supplied. This is mentioned here, as this definition is drafted without clear guidance in the OpenAPI specification for the handling of roles and scopes.

**5.2 Exceptions to Architectural Cross-Cutting Specifications**

**5.2.1 Privacy (REQUIRED)**

In general, the Consent Building Block shall follow the authentication and authorisation requirements as laid out in the [Govstack architecture](https://govstack.gitbook.io/specification/v/1.0/security-requirements/4-security-management). For clarity, Consent Building Block's API endpoints are invoked with a client-supplied API key which MUST defer to the Identification and Verification Building Block in order to verify the role and/or scope of the API key matches the API endpoint to which it is supplied. This is mentioned here, as this Definition is drafted without clear guidance in the OpenAPI spec for the handling of roles and scopes.

**5.3 Standards**

**5.3.1 ISO 8601**

All dates should follow ISO 8601.

**5.3.2 RFC 7159**

RFC 7159 - The JavaScript Object Notation (JSON).

**5.3.3 OpenAPI**

OpenAPI Version 3.1.0.

**5.3.4 REST API Design Guidelines Part 1**

RESTful APIs follow TM Forum Specification: “REST API Design Guidelines Part 1” (requirement derived from GovStack Architecture and Nonfunctional Requirements).