THE UNITED REPUBLIC OF TANZANIA

PRESIDENT'S OFFICE,
PUBLIC SERVICE MANAGEMENT

e-GOVERNMENT AGENCY

e-GOVERNMENT INTEGRATION ARCHITECTURE – STANDARDS AND TECHNICAL GUIDELINES

Document Number eGA/EXT/ITA/001

Issued by eGovernment Agency - November 2017

THE UNITED REPUBLIC OF TANZANIA

PRESIDENT'S OFFICE,
PUBLIC SERVICE MANAGEMENT

e-GOVERNMENT AGENCY

Document Title

eGovernment Integration Architecture – Standards and Technical Guidelines

Document Number eGA/EXT/ITA/001

APPROVAL	Name	Job Title/ Role	Signature	Date
Approved by	Dr. Jabiri Bakari	Chief Executive Officer		

Table of Contents

1.0 OVERVIEW

1.1. Introduction

The e-Government Agency (eGA) is established under the Executive Agencies Act No.30, 1997, Cap. 245 as a semi-autonomous Institution under President's Office Public Service Management. eGA is charged with the mandate of providing coordination, oversight and provision of e-Government initiatives and enforcement of e-Government standards to Public Institutions. In executing its duties, eGA shall implement and maintain coordinated government operations for Information and Communication Technology (ICT) that include the formulation of standards and guidelines to effectuate the purposes of the Agency.

To realize the vision of e-Government in Tanzania and successfully implement eGovernment Strategy, it is of paramount importance that **"e-Government Standards and Guidelines"** are formulated. The e-Government Standards and Guidelines' aim is to assist in the delivery of more consistent and cohesive services to citizen and support the more cost effective delivery of ICT services by Government. A worldwide agreeable practice for conducting Government wide eGovernment analysis, design, planning and implementation, using a holistic approach at all times, for the successful development and execution of eGovernment Strategy is known as **"eGovernment Enterprise Architecture".** The e-Government Standards and Guidelines Structure is hereby designed to cover most requirements of eGovernment Enterprise Architecture. This means that eGovernment Enterprise Architecture is incorporated in "eGovernment Standards & Guidelines".

Management of e-Government Standards and Guidelines requires categorisation. There are **nine categories/areas** covering all aspects of eGovernment. The **sixth** area is **eGovernment Integration Architecture.** The Integration Architecture defines the blueprint for Public Institutions to integrate their applications such that real-time seamless information exchange across the Government is enabled. Definition of integration architecture is incremental to meet specific Public Institution integration requirements for data sharing. In summary, the Integration Architecture establishes a comprehensive and uniform process for preparation, review, approval, deletion and documentation of Integration Architecture for use in information and communication technology management by Public Institutions.

The Integration Architecture Standards and Technical Guidelines document has been derived from the e-Government Enterprise Architecture as referred in *e-Government Architecture Vision - Standards and Technical Guidelines (eGA/EXT/AVS/001).*

1.2. Rationale

The Integration Architecture acts as an enabler for the e-Government Interoperability and addresses the convergence of the core infrastructure and supporting applications. Integration Architecture allows for the seamless exchange of information, reuse of data models and interchangeability of data across systems.

1.3. Purpose

In line with the above rationale, Integration Architecture helps to eliminate patchwork of ICT solutions in different Public Institutions such as those systems that are unable to "talk" or exchange data. It brings in the ability to effectively interconnect, collaborate, access and facilitate data integration in order to allow communication between different Public Institutions (G2G, G2C, G2E and G2B). Eventually this will ensure that the defined Integration Architecture Standards and Technical Guidelines is adopted across the Public Institutions.

1.4. Scope

This document applies to all Public Institutions. The Public Institution Accounting Officer (Head of Institution), Head of ICT Departments, Database Architects, Application Architects, Application Developers, Business Analysts and System Engineers shall be responsible for ensuring the effective implementation of these specific standards and technical guidelines associated with Integration Architecture within their respective Institutions.

2.0 e-GOVERNMENT INTEGRATION ARCHITECTURE

2.1. e-Government Integration Architecture Reference Framework

The Technical Reference Model (TRM) supports and enables the delivery of Integration Reference Model service components and capabilities and provides a foundation to advance the re-use and standardization of technology and service components from a government-wide perspective. Aligning ICT capital investments to the TRM leverages a common, standardized vocabulary allowing cross departmental discovery, collaboration, and interoperability. Public Institutions will benefit from economies of scale by identifying and re-using the best solutions and technologies to support their business functions, missions and target architecture. The TRM will continue to evolve with the emergence of new technologies and standards.

The TRM has been structured hierarchically as:

1. Service Area – Each Service Area aggregates the standards and technologies into a lowerlevel functional area. Each Service Area consists of multiple Service Categories and Service Standards.
2. Service Category – Each Service Category classifies lower levels of technologies and standards with respect to the business or technology function they serve. In turn each Service Category is comprised of one or more service standards.

3. Service Standards – They define the standards and technologies that support a Service Category. To support Public Institutions mapping into the TRM, many of the Service Standards provide illustrative specifications or technologies as examples.

The TRM for the Government is depicted in Figure I:

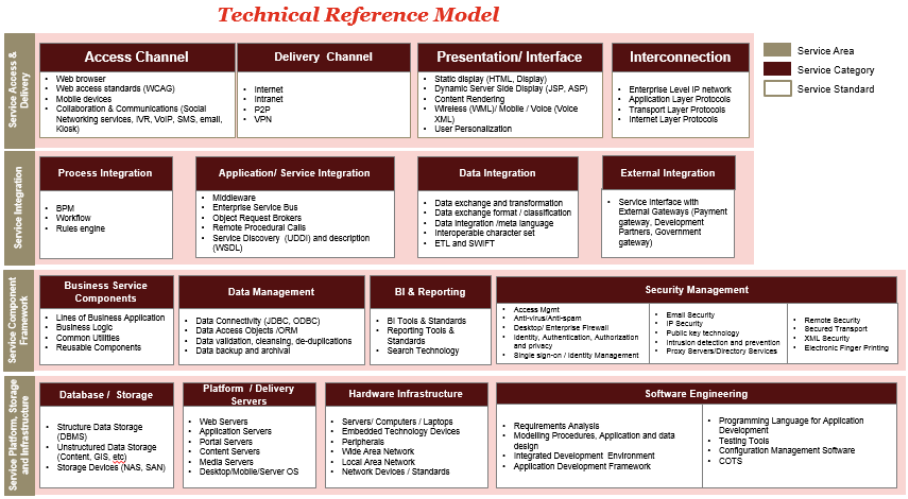


Figure I: Technical Reference Model

Public Institutions will standardise their TRM under 4 service areas:

i. Service Access and Delivery - This service area refers to the collection of standards and specifications to support external access, exchange and delivery of Service Components or capabilities.

ii. Service Interface and Integration - This service area refers to the collection of technologies, standards, and specifications that govern how Public Institutions shall interface both internally and externally with a service component. This area

also defines the methods by which components shall interface and integrate with backoffice/legacy assets.

iii. Service Component Framework - This service area refers to the underlying foundation, technologies, standards, and specifications by which Service Components are built, exchanged, and deployed across Distributed or Service-Orientated Architectures.

iv. Service Platform, Storage and Infrastructure - This service area refers to the collection of delivery and support platforms, infrastructure capabilities and hardware requirements to support the construction, maintenance, and availability of a Service Component or capabilities.

Deriving from the TRM is the Service Oriented Architecture (SOA) Reference Architecture - logical view which recommends the Reference Architecture based on SOA principles that must be adopted by all Public Institutions for new ICT initiatives. The SOA Reference Architecture - Middleware View depicted in Figure II provides the underlying middleware and infrastructure services required to build a SOA based solution.

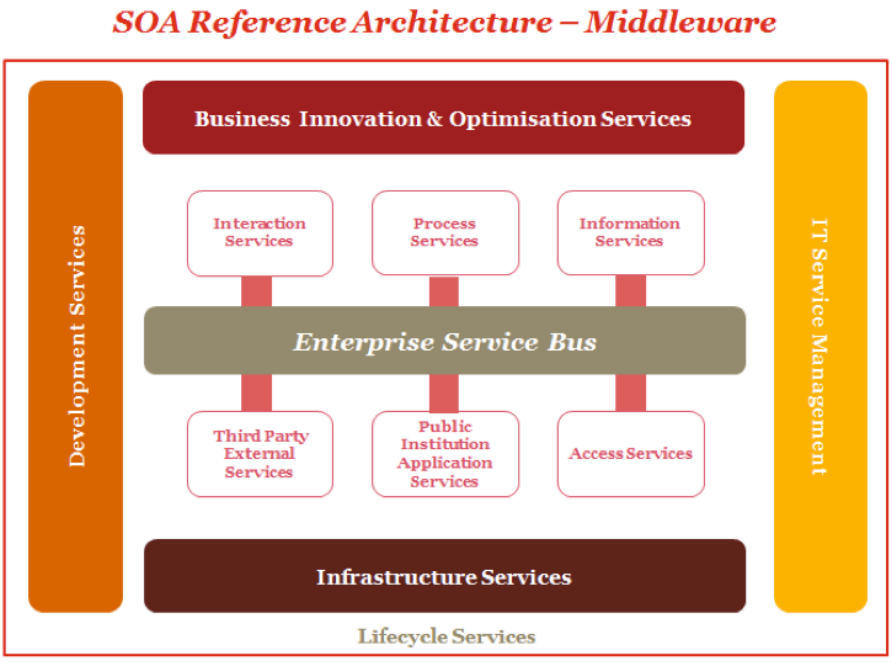


Figure II: SOA Reference Architecture – Middleware

The reference framework for functionalities of the various core and supporting services of the above SOA Reference Architecture are provided in Table I:

Table I: Functionalities of the various core and supporting services of SOA Reference

Core Services	Description
Interaction Services	i. Interaction services are a category of services that provide the presentation logic of the business design. These services are components that support the interaction between applications and end users. ii. Collaboration services also can be categorized as interaction services as they also provide a means for users to interact with the solution
Process Services	Process services are a category of services that include various forms of compositional logic, the most notable of which are business process flows, business state machines, business rules, and decision tree processing.
Information Services	Information services are a category of services that contain the data logic of business design. This includes the data access, data composition, data flow, canonical data model, and the data and metadata standards to be considered for information sharing.
Public Institution Application Services	These are the category of services that implement core business logic of the Public Institution applications.
Access Services	Access services are a category of services that are dedicated to integrating legacy applications and functions into the SOA solution. Access service implementation could be simple wrappers or complex adapters around the legacy applications to facilitate integration across heterogeneous platforms.
Third Party External Services	These are a category of services that capture the interoperable policies and semantics that partners with and external third parties should conform with to interact with the government and the avail the services. It projects a view of the government services and businesses to the external parties and controls the interaction with them.

Infrastructure Services	Infrastructure services are the category of services that form the core of the information technology environment for hosting SOA applications.
Enterprise Service	This is the category of service that assumes the responsibility for binding service consumers with service providers.
Bus (ESB)	i. ESB should be transparent to the service consumer in the SOA solution. ii. Implementations of the service connectivity services support interconnectivity and Host Mediations, i.e., logic that may perform message transformation, intelligent routing, augmented functionality (such as logging or auditing) to enable the interconnectivity of services. iii. Service connectivity services are a mix of domain-neutral (messaging products and ESBs available from many vendors), and domain-specific (the implementations of adapters needed from existing services and operational systems into the ESB).
Development Services	i. Development services are a category of services that encompass the entire suite of architecture tools, modelling tools, development tools, visual composition tools, assembly tools, methodologies, debugging aids, instrumentation tools, asset repositories, discovery agents, and publishing mechanisms needed to construct an SOA-based application.
	ii. Some development tools have a built-in mechanism for modularizing and plugging in tool services, thus encouraging the construction of the development tools as services following many of the same principles promoted by SOA.
IT Service Management	i. Tools and templates to align IT Service Delivery and IT Service Support with the needs of Public Institutions.

Life Cycle Services	ii. Life cycle services are a category of services that support managing the life cycle of SOA solutions and all of the elements that comprise them across development and management, ranging from strategy to infrastructure.

In Based on the above SOA reference architecture components, the standards and guidelines for Integration Architecture have been defined for adoption by Public Institutions.

As defined in *eGovernment Business Architecture - Standards and Technical Guidelines (eGA/EXT/BSA/001),* the need for a service delivery gateway is of outmost importance to help create an integrated Government connecting Public Institutions applications, making services accessible to the citizens and ensure the efficiency, transparency and reliability of such services.

On the other hand, in order to achieve information exchanged between the service provider and consumer, the use of Enterprise Service bus is vital. ESB provides a middleware infrastructure that is process/Model driven, loosely coupled, support integration of heterogeneous systems, based on open standards, helps in rapid development, assembly and deployment of services, ease of maintenance and improved business visibility. The diagram in Figure III represents a typical ESB conceptual architecture.

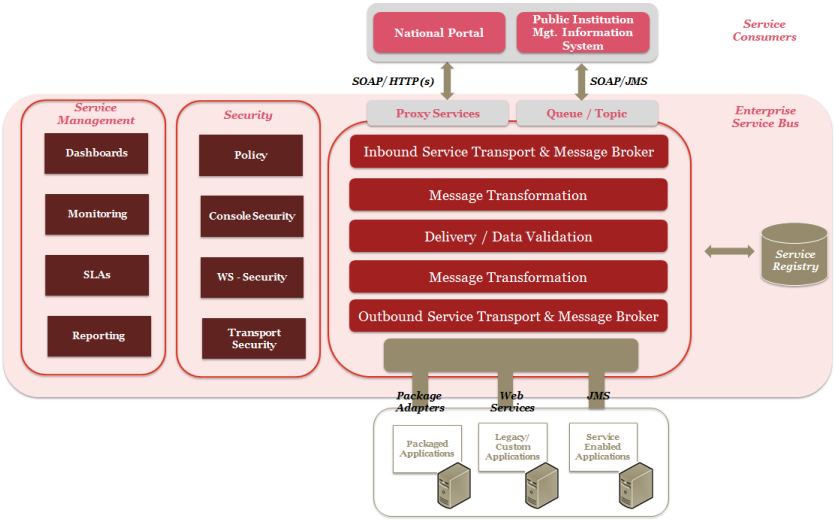


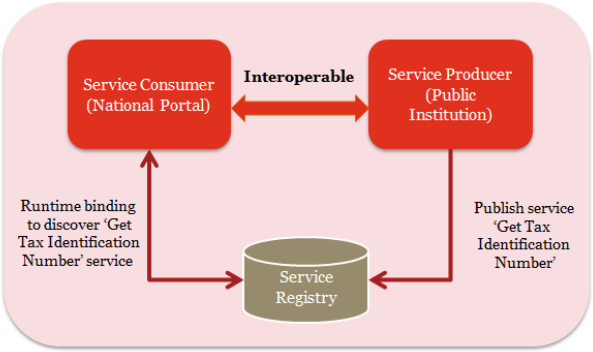
Figure III: Enterprise Service Bus

The ESB is a dynamic and configurable message and service broker. The reference framework for the ESB capabilities is provided in Table II:

Table II: ESB Features & Capabilities

ESB Features	ESB Capabilities
Message brokering between heterogeneous environments	i. Support asynchronous, synchronous, publish and subscribe messaging. ii. Support web services: such as SOAP, WSDL and UDDI, as well as emerging standards such as WS-Reliable Messaging and WS-Security.
	iii. Support multiple message formats including SOAP, SOAP with attachments, XML, structured non-XML data, raw data, text, and e-mail with attachment.
Heterogeneous transports between service end points	i. Support multiple protocols such as file, FTP, HTTP(s), multiple JMS providers (SOAP/JMS), RMI, web services (SOAP/HTTPs), MQ, CORBA, DCOM, and email (POP, SMTP, IMAP).
Message transformation to enable the service consumer to talk to the producer	i. XML to XML message transformation from the proprietary service consumer application specific message structure to common canonical data model structure.
Content-based routing	i. Publish and subscribe routing across multiple types of sources and destinations. ii. Support for both point-to-point and one-to-many routing scenarios, enabling request-response and publish subscribe models.
Service Management and Monitoring	i. Service monitoring, logging, and auditing with search capabilities. ii. Capture of key statistics for message and transport attributes, including message invocations, performance,

	errors, volume, and SLA violations.
High availability	i. Support clusters and gathers statistics across the cluster to review SLA violations. ii. Simplify service provisioning. a) Deploy new versions of services dynamically through configuration. b) Migrate configured services and resources between design, staging and production. c) Support multiple versions of message resources that are incrementally deployed with selective service access through flexible routing.
Configurable policies driven security	i. Support the latest security standards for authentication, encryption-decryption, and digital signatures. ii. Support SSL for HTTP and JMS transports. iii. Support multiple authentication models.
Policy-driven SLA enforcement	i. Establish SLAs on a variety of attributes including throughput times, processing volumes, success/failure ratios of message processes, number of errors, security violations, and schema validation issues. ii. Initiate automated alerts or enables operator-initiated responses to policy violations using flexible mechanisms including e-mail notifications, triggered JMS messages, triggered integration processes with a JMS message, web services invocations with a JMS message, or administration console alerts.
Platform-neutral	Connect to any technology in the enterprise, e.g. Java, .Net, mainframes, and databases.
Service registry	SOA requires services to be coarse-grained, loosely coupled, and standards-based. As services are developed and deployed there must be a catalogue of services available for architects, developers, operations, and business managers.



The above diagram illustrates the architecture of the service registry. The service producer publishes the service to the service registry which is leveraged by the service consumer for runtime binding. The registry also acts as the system of record for the business policies that it enforces at runtime. The service registry shall provide the following capabilities:

i. Core services, including replication, UDDI data store, and security.

ii. Information services, including data validation, SOA mappings, advanced classification, and business data access service.

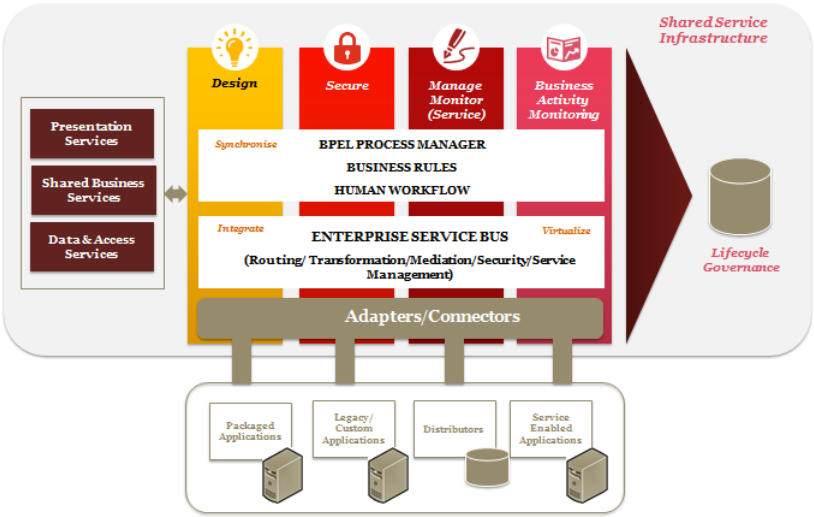
iii. Life cycle services, including approval and change management, change notification, business service discovery, and QoS management.

iv. Web-based business service console for configuration.

v. Platform-independent open architecture that interfaces with leading enablement, management and security products.

In the Enterprise Service Bus, will create an integrated Government which is connected to all Public Institutions to facilitate real-time data sharing. Whilst adopting the SOA Reference Architecture, below is the proposed SOA Conceptual Architecture:

Figure IV: SOA Conceptual Architecture



Finally, the *eGovernment Infrastructure Architecture - Standards and Technical Guidelines (eGA/EXT/INF/001)* provides reference framework for service type, service component and service component capabilities.

2.2. e-Government Integration Architecture Standards

2.2.1. Table III provides principles under which the eGovernment Integration Architecture is designed. Institutional integration architecture component of enterprise architectures should also be designed basing on these principles (There is an example in the Appendix).

Table III: eGovernment Integration Architecture Design Principles

Principle #1	***Interoperability***
Rationale	i. The SOA Reference Framework eliminates patchwork of ICT solutions in different Public Institutions such that those systems are able to "talk" or exchange data. ii. Interoperability allows seamless exchange of information, reuse of data models and inter-changeability of data across systems. iii. Brings in the ability to effectively interconnect, collaborate, access and facilitate data integration in order to communicate between G2G, G2C, G2E and G2B.
Implications	i. Policies defined should reinforce and standards selected should facilitate interoperability. ii. The Government will identify common components (including existing Government policies, standards, application, technology etc. wherever relevant) across the interoperability domain and define policies, standards, and procedures to ensure reusability of artefacts. For e.g. defining data structure, data sets at a national level etc. iii. Public Institutions will choose standards that will enable more choice and reduce the administrative burden.
Principle #2	***Confidentiality***
Rationale	Confidential information and data are properly classified and adequately protected. Privacy cannot be guaranteed by technical standards alone, it has to have process, inter-organisational agreements, cyber laws etc. in place to enforce it.
Implications	Public Institutions will guarantee the privacy of information with regard to citizens (e.g. health records), business (e.g. Public Institution statistics) and government (e.g. confidentiality agreements) and enforce the legally-defined restrictions on access and dissemination of information.

Principle #3	***Open standards based***
Rationale	i. Adherence to open standards will be promoted
Implications	i. The attributes of open standards such as platform independence, vendor neutrality and ability to use across multiple implementations and the model for establishing open standards are what will allow for sustainable information exchange, interoperability, flexibility, data preservation & and greater freedom from technology and vendor lock-in. ii. Adoption of open standards will facilitate storing of electronic national records and data using open data file formats.
Principle #4	***Government service delivery gateway based on Enterprise Service Bus***
Rationale	i. The use of ESB promotes loose coupling, support integration of heterogeneous systems, support adherence to open standards. ii. ESB enables rapid development, assembly and deployment of services, ease of maintenance and improved business visibility
Implications	i. The Enterprise Service Bus (ESB) will be the public Application Programming Interface (API) for the underlying implementation of the National Service Delivery Gateway. ii. The ESB will be available as a resource for any service components in the Public Institution. There should be loose coupling between the service and its underlying layers with the service layer acting as a façade the layer below it.
Principle #5	***Web services for information exchange and granular service.***
Rationale	Web Services will be used between the service layers. Granularity of the services composed in the ESB should not be too fine to promote a huge number of unmanageable services, where change in one results in a cascaded set of changes with the coherent others
Implications	Public Institutions will comply with Government standards for web service specifications of security, interoperability, reliability, etc.

2.2.2. The SOA design principles in Table IV are to be used while designing SOA based services:

Table IV: SOA design principles

Principle #1	***Standardized Service Contract***
Rationale	The design principle states that the services within the same service inventory are in compliance with the same contract design standards.
Implications	Public Institutions will ensure that service contracts are both optimized, appropriately granular, and standardized to ensure that the endpoints established by services are consistent, reliable, and governable. This design principle essentially advocates "contract first" design for services.
Principle #2	***Service Loose Coupling***
Rationale	This principle advocates the creation of a specific type of relationship within and outside of service boundaries, with a constant emphasis on reducing ("loosening") dependencies between the service contract, its implementation, and its service consumers.
Implications	Public Institutions will promote independent design and evolution of a service's logic and implementation while still guaranteeing baseline interoperability with consumers that have come to rely on the service's capabilities.
Principle #3	***Service Abstraction***
Rationale	The design principle states that the service contracts contain only essential information, and information about services is limited to what is published in service contracts.
Implications	The principle emphasizes the need to hide as much of the underlying details of a service as possible. Doing so directly enables and preserves the previously described loosely coupled relationship.

Principle #4	*Service Reusability*
Rationale	The design principle of Service Reusability states that services contain and express agnostic logic and can be positioned as reusable Enterprise Resources.
Implications	The principle focuses on ensuring that service logic is designed to be robust and generic, and emphasizes the positioning of services as enterprise resources with agnostic functional contexts.
Principle #5	*Service Autonomy*
Rationale	The design principle states that the services exercise a high level of control over their underlying runtime execution environment.
Implications	For services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources.
Principle #6	*Service Availability*
Rationale	The design principle states that the services minimize resource consumption by deferring the management of state information when necessary.
Implications	The management of excessive state information can compromise the availability of a service and undermine its scalability potential. Services shall be therefore ideally designed to remain stateful (available) only when required.
Principle #7	*Service Discoverability*
Rationale	The design principle states that the services are supplemented with communicative metadata by which they can be effectively discovered and interpreted.
Implications	For services to be positioned as ICT assets with repeatable Return on Investment (ROI) they need to be easily identified and understood when opportunities for reuse present themselves. The service design therefore needs to take the "communications quality" of the service and its individual capabilities into account, regardless of whether a discovery mechanism (such as a service

	registry) is an immediate part of the environment.
Principle #8	***Service Composition***
Rationale	The design principle states that the services are effective composition participants, regardless of the size and complexity of the composition.
Implications	Complex service compositions place demands on service design that need to be anticipated to avoid massive retro-fitting efforts. Services shall be capable of participating as effective composition members, regardless of whether they need to be immediately enlisted in a composition.
Principle #9	***Service- Orientation and Interoperability***
Rationale	A fundamental goal of applying service-orientation is that a level of intrinsic interoperability is established as a common and expected service design characteristic.
Implications	Supports each of the eight principles listed above or contributes to interoperability in some manner. Below are just a few examples: i. When Service contracts are be standardized they will guarantee a baseline measure of interoperability associated with the harmonization of data models. ii. Reducing the degree of service coupling will foster interoperability by making individual services less dependent on others and therefore more open for invocation by different service consumers. iii. Abstracting details about the service will limit all interoperation to the service contract, increasing the long-term consistency of interoperability by allowing underlying service logic to evolve more independently. iv. Designing services for reuse will imply a high-level of required interoperability between the service and numerous potential service consumers. v. By raising a service's individual autonomy its behaviour will become more consistently predictable, increasing its reuse

	potential and thereby it's attainable level of interoperability. vi. Through an emphasis on stateless design, the availability and scalability of services increase, shall allow them to interoperate more frequently and reliably. vii. Service Discoverability simply will allow services to be more easily located by those who want to potentially interoperate with them. viii. Finally, for services to be effectively composed they must be interoperable. The success of fulfilling composed requirements will be tied directly to the extent to which services are standardized and cross-service data exchange is optimized.

2.2.3. Public Institutions will adhere to Figure III: Enterprise Service Bus for the implementation of the Service Delivery Gateway Middleware Platform.

2.2.4. In the Enterprise Service Bus, Public Institutions will define their SOA based enterprise service integration platform by leveraging the SOA Conceptual Architecture as illustrated in Figure IV: SOA Conceptual Architecture.

2.3. eGovernment Integration Architecture Technical Guidelines

2.3.1. The following are guidelines for Application Integration:

i. ESB will be used as the public middleware for the underlying implementation of the Service Delivery Gateway to facilitate data exchange across the Public Institutions through a common interoperable platform leveraging standard web service based protocols.

ii. Integration between applications through the integration platform will be facilitated through combination of the following mechanisms:
   a. Real time synchronous integration
   b. Web service based real-time asynchronous integration
   c. Message based batch integration

iii. All interactions between service consumers (citizens and businesses) and service providers (Public Institutions) will be only through the National Enterprise Service Bus gateway. Legacy applications residing at Public Institutions will also

offer their services to various other consumers connected to the Enterprise Service Bus.

iv. Each Public Institution will register their services in the centralized ESB service registry. The ESB will act as a proxy for invoking these registered remote services of the Public Institution related business applications. . The web services registered in the service registry could be discovered by any citizen or business that needs access to the service.

v. Public Institution will leverage ESB to handle specific data transformations, content based routing and filtering and service orchestrations. The ESB shall facilitate real time and near real time synchronization and co-ordination of inter departmental tasks, tracking all online transactions of the Public Institutions. The ESB platform shall provide robust security and audit feature for each service invocation that will result in better tracking (auditing) and will enforce government control through complete audit logs and time stamping of transactions.

vi. All Public Institutions exposing services through the ESB will adhere to the data exchange formats and XML schema definition.

vii. The ESB will provide the necessary connectors to enable interfaces with the Public Institutions applications. The ESB shall have the capability to subsequently add additional functionality to support shared common services like authentication, payment gateway interfaces, short messaging services, instant messaging services etc.

viii. In Public Institutions with no immediate real-time integration needs, batch data load could be considered for quick data transformation and load.

ix. To illustrate the information exchange across the proposed service delivery gateway, some end-to-end use case scenarios have been depicted in Appendix – Illustration No1 to demonstrate how the above ESB topology could be leveraged as the integration platform for information exchange across Public Institutions.

2.3.2. The following are guidelines for Data Integration:

i. Follow established common data and Meta data standards/ format to access, share and integrate data.

ii. To enhance data integration identify the enterprise core common data entities which will be used across the institution for data sharing e.g. Citizen's demographic profile, name, address etc.

iii. Define the canonical data model which represents the common data entities identified in the previous step, their data attributes, the data types and the relationship between these data entities.

iv. Establish the data sharing services and define the SOA service specification required to support the data sharing needs. The model is created to help understanding the data standard, what format it will be in if they request it and how it can be accessed / exchanged.

v. Addressing security and privacy issues has been a major hindrance to data sharing in some areas of government. To identify and implement methods for addressing security and privacy requirements by making use of security guidance refer to Security Architecture - Standards and Technical Guidelines (eGA/EXT/ISA/001)

vi. Consider data quality, confidentiality and integrity as the data is transmitted from the source to other environments.

vii. The Government of Tanzania shall define a Government wide XML schema data sharing across the interoperability framework which will be based on the above common data specification.

viii. All Public Institutions that expose their Government services such as e-Services shall adhere to the recommended common data exchange specification as defined in the Government Data XML Schema and the exchange package (or web service contract definition) to enable seamless information flow.

ix. Public Institutions shall establish data sharing agreements. An evaluation has to be performed on the required changes in rules at all appropriate levels of government that will be needed to support data sharing agreements.

2.3.3. The following are guidelines for Web Service Design:

i. The first step is to perform the basics of design planning by assessing what is in place today, what the goals are, and what technology has to be used in order to position the applications for future growth. The following list provides a high-level view of the planning tasks that should be performed:

1.a) Review the standards used in the development and design phase. Web services can be based on a variety of programming models. Start by identifying how the existing Web services are designed as well as the APIs, standards, and specifications that were part of the design.

1.b) Identify goals by considering: what needs to be accomplished by using Web services, identify applications and business logic that has to be made available

as a service. Consideration shall also be given to the existing services that need to be migrated to newer technology.

1.c) Determine how Web services fit into the current topology, applications, and programming model, how the current Web services process requests on the server and how the clients manage and use the Web service. These factors will be considered when planning for new or migrated Web services.

1.d) Design the Web services for non-functional requirements to fit the business solution. In other words, Web services will be designed for reliability, availability, manageability, and security. For example, Web services might be required to process a transaction in a reasonable amount of time at all hours of the day and provide users with optimal security, such as authentication mechanism. Such performance implications will be factored during the design.

1.e) Decide which development and implementation tools to use, variety of manual development and implementation tasks for implementation of existing Web services or development of new services can be used. Consider JavaBeans implementation or from an Enterprise JavaBeans (EJB) module, different tasks can be chosen in respect to the available resources. Also consider the use of assembly tools to complete development and implementation tasks.

1.f) Select the best suited required runtime environment (Application server runtime topology) based on the functional and non-functional requirements.

1.g) Adhere to the Web Services-Interoperability (WS-I) Basic Profiles - The Basic Profile defines how a selected set of specified Web Services technologies, such as messaging and discovery, should be used together in an interoperable manner.

2.3.4. The following are guidelines for Web Service Development:

i. Make use of good practices for simple data types and avoid use of Java collection classes and similar complex data types, as there might not be proper counterparts on the client side.

ii. Design Web Service applications for course-grained service with moderate size payloads, avoid fine grained web services. Course-grained Web services allow the Web service to return more data in response to a single request, rather than having multiple requests to retrieve smaller portions of data. Working with coarser grained services also allows a single service to be reused, instead of creating multiple fine-grained services.

iii. Avoid deep nesting of XML structures - Since parsing of deeply nested XML structures increases the processing time, deeply nested compound data types should be avoided. This also increases comprehension time of the data type itself.

iv. Make use compressed XML for sending the messages over the network.

v. Make use of Web services caching as provided by the platform.

vi. Consider using asynchronous calls to invoke web services which involve complex and long running processes. An asynchronous invocation of a Web service sends a request to the service endpoint and then immediately returns control to the client program without waiting for the response to return from the service. Give consideration to the length of the target service invocation, adjust client timeout, and handle errors appropriately in order to avoid losing control of the application flow.

vii. Choose correct service and encoding styles for WSDL binding customization and make use of document/literal wrapped WSDL style for the greatest level of interoperability.

viii. Externalize web services security and management from the applications you build. Instead of coding security logic in the application, implement declarative security and management through predefined policies.

2.3.5. The following are Web Service Performance considerations guidelines:

i. Reduce the Web services requests by using a coarse grain service rather than several simple fine grain services (as recommended in the SOA design patterns guidelines earlier).

ii. Design the WSDL file interface to limit the size and complexity of SOAP messages.

iii. Use the document/literal style argument when the WSDL file is being generated.

iv. Leverage the caching capabilities offered by the underlying platform.

v. Test applications for performance and then monitor them to ensure that performance goals are being met and optimise on the performance monitoring infrastructure of the underlying platform.

2.3.6. For Web Service Security guidelines refer to *eGovernment Security Architecture - Standards and Technical Guidelines (eGA/EXT/ISA/001).*

3. IMPLEMENTATION, REVIEW AND ENFORCEMENT

3.1. This document takes effect once approved in its first page.

3.2. This document is subject to review at least once every three years.

3.3. Any exceptions to compliance with this document should be approved in writing by Chief Executive Officer (CEO) of e-Government Agency.

4. GLOSSARY AND ACRONYMS

4.1 Glossary

None

4.2 Acronyms

Abbreviation	Explanation
BCP	Business Continuity Planning
API	Application Programming Interface
ESB	Enterprise Service Bus
ICT	Information and Communication Technology
SOA	Service Oriented Architecture
TRM	Technical Reference Model
UDDI	Universal Description, Discovery and Integration
WSDL	Web Services Description Language
WS-I	Web Services - Interoperability
XML	eXtensible Markup Language

5. RELATED DOCUMENTS

5.6. **eGovernment Integration Architecture - Standards and Technical Guidelines (eGA/EXT/ITA/001)**

5.7. **eGovernment Infrastructure Architecture - Standards and Technical Guidelines (eGA/EXT/IRA/001)**

5.8. **eGovernment Security Architecture - Standards and Technical Guidelines (eGA/EXT/ISA/001)**

5.9. **eGovernment Architecture Processes and Governance - Standards and Technical Guidelines (eGA/EXT/PAG/001)**

6. DOCUMENT CONTROL

Version	Name	Comment	Date
Ver. 1.0	eGA	Creation of Document	February 2016
Ver. 1.1	eGA	Alignment with eGovernment Guideline 2016	December 2016

APPENDIX

Illustration No.1 Integration Architecture for a sample use case scenario- Online Voter's Registration Process

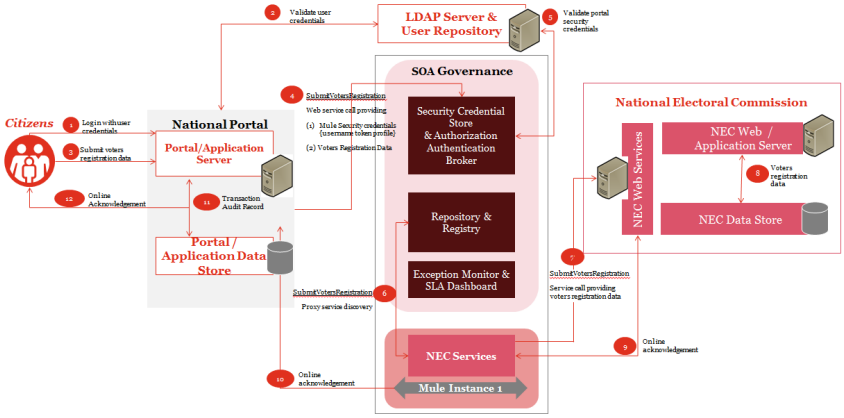


Figure A1: Online Voter's Registration Process

Table A1: Voter's Registration Process Steps

S.No.	Steps
1.	Citizen logins to the National Portal with user credentials. The citizen would be required to register online with the national portal prior to login through the - User Registration option. The online user credentials provided as part of the User Registration process will be required by the citizen for login to the national portal.
2.	i. The national portal authenticates the user credentials against the user repository in LDAP. ii. If authentication fails, the portal returns an error. iii. If authentication is successful, the user is allowed to login to the national portal to avail the e-Service.

3.	i. The user chooses the National Electoral Commission (NEC) - Voters Registration eService and opts for the online application for Voter's Registration. ii. The user will be presented with the online voters registration form. iii. The user fills the online form with the relevant information required and submits the online form.
4.	i. After submission, the portal will convert the online form data captured into XML message format which is sent to a listening servlet running in the portal server. ii. The servlet will process the request and in turn invoke the - Submit-VotersRegistration web service to submit the information. The user name token for the portal proxy user defined will be appended as message header. iii. The form data and the citizen user credentials will be passed as the web service input requests and will be part of the message body. iv. The portal instance deployed in the National Electoral Commission will try to establish connection to the central Mule based ESB infrastructure with relevant authentication security credentials to invoke the - SubmitVotersRegistration web service.
5.	i. Mule will authenticate the user ID and Password provided in the user name token profile passed by the portal instance against the LDAP user repository. ii. If the user name token profile is not valid Mule will through an exception and return an error message back to the portal. iii. If the user name token is valid, Mule will check if the user profile is authorized to access the respective web service. If the user profile is not authorized to access the web service, Mule will return an error message back to the portal. iv. If the user name token profile is valid and is authorized to access the web service, Mule will continue further to process the request as mentioned in Step 6.

6.	i. The Mule - NEC Services which acts as the proxy for the underlying web service - SubmitVotersRegistration deployed in NEC server will be deployed in Mule instance 1. This proxy service will be registered in the Registry. ii. Mule will invoke the - NEC services deployed in Mule instance 1 through service discovery of the already registered - SubmitVotersRegistration service in the registry.
7.	i. Mule proxy - NEC Service will establish connection with the department specific NEC web server that hosts the voters registration system & the SubmitVotersRegistration web service. ii. The SubmitVotersRegistration web service hosted in NEC server will be invoked. The online request initiated from the national portal will reach the department server through the Mule based ESB service delivery gateway.
8.	The NEC SubmitVotersRegistration web service will process the form data provided by the citizen and store the data in the data store.
9.	The online application acknowledgement with a reference number / application number will be returned as part of the response back to Mule - NEC Service.
10.	The Mule NEC Service deployed in Mule instance 1 will return the response provided by the NEC service back to the portal server.
11.	The audit details for the transaction are captured in the portal data store.
12.	The portal displays the response online to the citizen providing the reference number / application registration number for tracking the status of the online application.

Issued by eGovernment Agency - Novemebr 2017