Consular Systems and Technology Modernization Patterns of Design & Architecture

Document Version 2.0

**Dated** : February 19, 2016

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| ***Date*** | ***Version*** | ***Description*** | ***Author*** |
| 11/06/2015 | 1.0 | Initial submittal to CST by EA team | Haywood Barney (Lead) Supriya Ghosh Moazam Gilanian Kalvin Lee  Corrine Marr Sameer Nayak |
| 01/09/2016 | 2.0 | Expanded the scope of pattern practices ranging from the classic OOP to SOA, service security and Cloud computing by the SEI team. | Michael Steele  Michelle Cho  Peter Stark  Burt Sun |
|  |  |  |  |
|  |  |  |  |

This document is prepared in a manner where each section is considered a separate module. Periodic update and version management will be tracked separately for each section.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial Approvals and Concurrences | | | | |
| **Authority** | **Date** | **Comments** |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name: ThyNgon Tran |  |  |  | Approve |
|  | Disapprove |
| CST/SEI Enterprise Architect |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name: Michael J. Meccia |  |  |  | Approve | |
|  | Disapprove | |
| CST/SEI Branch Chief |  |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name: Sandeep Maripuri |  |  |  | Approve | |
|  | Disapprove | |
| CST/SII Division Chief |  |  | |

Table of Contents

[1. Architecture & Design Patterns 3](#_Toc442353817)

[1.1 Classic Design Patterns in Object Oriented Programming or Coding 3](#_Toc442353818)

[1.2 Design Patterns in SOA with REST 5](#_Toc442353819)

[1.3 Design Patterns in Service Security 6](#_Toc442353820)

[1.4 Design Patterns in Cloud Service & Storage Security 9](#_Toc442353821)

[1.5 Design Patterns in Cloud Computing – Sharing, Scaling and Elasticity 13](#_Toc442353822)

[1.6 Design Patterns in Cloud Access Management 15](#_Toc442353823)

[1.7 Architecture Patterns ( Supplemental Listing ) 17](#_Toc442353824)

[1.8 Services and Systems (Infrastructure) Patterns 20](#_Toc442353825)

[1.9 Data Patterns 27](#_Toc442353826)

[1.10 Information Assurance (IA) Patterns 30](#_Toc442353827)

[2. Reference Architecture Principles & Rules 35](#_Toc442353828)

[2.1 CA Architecture Principles and Associated Rules 36](#_Toc442353829)

[3. Alphabetical List of Patterns 42](#_Toc442353830)

Table Categories of Design & Architecture Pattern

Table 1‑1. Selected Design Patterns in Object Oriented Programming or Coding 3

Table 1‑2. Selected Design Patterns in REST Web Service 5

Table 1‑3. Selected Design Patterns in Service Security 6

Table 1‑4. Selected Design Patterns in Cloud Service & Storage Security 9

Table 1‑5. Selected Design Patterns in Cloud Computing – Sharing, Scaling and Elasticity 13

Table 1‑6. Selected Design Patterns in Cloud Access Management 15

Table 1‑7. Architecture Patterns - Supplementary 17

Table 1‑8. Services and Systems (Infrastructure) Patterns 20

Table 1‑9. Data Patterns 27

Table 1‑10. Information Assurance Patterns 31

Tables of Principles and Rules

Table 2‑1-APR. CA Architecture Principles and Rules 36

Table 2‑1-DSP. Data Specific Principles 38

Table 2‑1-SIP. System/Service Interoperability Principles 38

Table 2‑1-SP. Security Principles 40

# Architecture & Design Patterns

It is human nature to think in patterns and to apply patterns to the things we do. The use of architecture patterns provide confidence that a situation that has been mastered using a specific pattern may be mastered again using the same pattern again in a similar situation. That is the value of developing architecture patterns for those aspects of the enterprise where the development of capability has been “mastered” and that pattern should be reused to situations where that capability applies. The use of patterns is a best practice in the IT community. The sections that follow capture the enterprise patterns across CA that provide a consiberable contribution to the architecture principles of reusability and interoperability.

The patterns are grouped using the same groupings as the principles and rules:

* Classic Design Patterns in OOP
* Design Patterns in SOA & REST
* Design Patterns in Service Security
* Design Patterns in Cloud Computing – Sharing, Scaling and Elasticity
* Design Patterns in Cloud Access Management
* Services and Systems (Infrastructure)
* Data
* Information Assurance
* Architecture Patterns’ Supplemental Listing

Not all patterns are fully described in the CST Modernization RA. This document simply references those patterns that are well known and will be captured within external sources.The following sections comprise many patterns tables that introduce and summarize each pattern that relates to one of the previously presented groupings. The patterns tables describes each pattern, graphically captures its context, and maps the applicable principle(s).

## Classic Design Patterns in Object Oriented Programming or Coding

**Table 1‑1. Selected Design Patterns in Object Oriented Programming or Coding‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Singleton |  | | * Ensuring that only one instance of a class is created and providing a global point of access to the object, the implementation involves a static member within the class, a private constructor and a static public method that returns a reference to the static member. | Singleton Implementation - UML Class Diagram | Creational |
| Adapter | * Convert the interface of a class into another interface clients expect. Adapter lets classes work together, that could not otherwise because of incompatible interfaces. | Adapter  Pattern Implementation - UML Class Diagram | Structural |
| Factory | * Creates objects without exposing the instantiation logic to the client, referring to the newly created object through a common interface | Factory Implementation - UML Class Diagram | Creational |
| Observer | Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. | Observer Design Pattern Implementation - UML Class Diagram | Behavioral |
| Iterator | Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation. The abstraction provided by the iterator pattern allows you to modify the collection implementation without making any changes outside of collection. | Iterator Implementation - UML Class Diagram | Behavioral |
| Decorator | Being capable of adding additional or removing existing responsibilities dynamically to an object, the pattern maintains a reference to a Component object and defines an interface that conforms to Component's interface. | Decorator Pattern Implementation - UML Class Diagram | Structural |

## Design Patterns in SOA with REST

**Table 1‑2. Selected Design Patterns in REST Web Service‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Reusable  Or Uniform Contract |  | | Limit tight coupling to a common, reusable technical contract that is shared by multiple services. The technical contract provides only generic, high-level functions that are less likely to be impacted when service logic changes. |  | SP1 |
| Content Negotiation | Allow the service capability to support alternative formats and representations by providing a means by which consumer and service can “negotiate” data characteristics at runtime. The pattern is most commonly applied via HTTP media types that can define the format and/or representation of message data. The media type of the data is decoupled from the data itself, allowing the service to support a range of media types. |  |  |
| Entity Linking | Services inform their consumers about the existence of related entities as part of the consumer's interactions with the services by including links in relevant response messages from the service. Service consumers are able to navigate from entity to entity by following these links, and accumulate further business knowledge along the way. | Entity Linking |  |
| Endpoint Redirection | Automatically redirect service consumers that attempt to access out-of-date service endpoints to the current service endpoints by including endpoint redirection as a feature of the service contract. |  |  |
| Lightweight Entity Endpoint | Expose data and functionality associated with business entities as a series of granular, lightweight endpoints, offering service consumer a range of functional granularity. |  |  |
| Idempotent Capability | Idempotency guarantees that repeated invocations of a service capability are safe and will have no negative effect. The design of an idempotent capability can include the use of a unique identifier with each request so that repeated requests (with the same identifier value) that have already been processed will be discarded or ignored by the service capability, rather than being processed again. |  |  |

## Design Patterns in Service Security

**Table 1‑3. Selected Design Patterns in Service Security‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Trusted Subsystem |  | | As consumer that accesses backend resources of a service directly can compromise the integrity of the resources and further lead to undesirable forms of implementation coupling, the service is designed in such a way that it uses its own credentials for authentication and authorization with backend resources on behalf of consumers. | Trusted Subsystem: Neither a malicious or non-malicious consumer can access the database directly. Only the service itself can access the database with its own credentials. | Service Loose Coupling |
| Exception Shielding | Potentially unsafe exception data is "sanitized" by replacing it with exception data that is safe by design before it is made available to consumers. The practice can be applied at design time by reviewing and altering source code or at runtime by adding dynamic sanitization routines. | Exception Shielding: Potentially unsafe information is sanitized by routines added to the service logic, thereby releasing only safe exception information to service consumers. | Service Abstraction |
| Message Screening | The service is equipped or supplemented with special screening routines that assume that all input data is harmful until proven otherwise through a number of checks to screen message content for malicious attempt. | Message Screening: Because the service logic is equipped with extra message screening routines, malicious or malformed data can still be detected and rejected before it has a chance to do harm. | Standardized Service Contract |
| Boundary Guard | An intermediate service is established at the perimeter of the private network as a secure contact point for any external consumers that need to interact with internal services. | Service Perimeter Guard: The perimeter service processes the attacker's message and upon determining its malicious intent, rejects it. This spares the underlying internal service from exposure and unnecessary security-related processing. | Service Loose Coupling and Service Abstraction |
| Brokered Authentication | Since the use of direct authentication can be impractical or even impossible when consumers and services do not trust each other or when consumers are required to access multiple services as part of the same runtime activity, an authentication broker with a centralized identity store assumes the responsibility for authenticating the consumer and issuing a token that the consumer can use to access the service. | Brokered Authentication: The consumer submits a request with credentials to the authentication broker (1), which the broker authenticates against a central identity store (2). The broker then responds with a token (3) that the consumer can use to access Services A, B, and C (4), none of which require their own identity store. | Service Composability |
| Data Confidentiality | The message contents are encrypted independently from the transport, ensuring that only intended recipients can access the protected data and that security is not compromised at any point of service intermediary. | Data Confidentiality: Data Confidentiality protects the message while in transit between services and while in the possession of unauthorized intermediaries. | Service Composability |
| Data Origin Authentication | A message can be digitally signed so that the recipient services can verify that it originated from the expected consumer and that it has not been tampered with during transit. Often asymmetric encryption and decryption algorithm, such as those specified in the XML-Encryption standard, is applied at the message level. | Data Origin Authentication: In this scenario, the attacker could be attempting to take a valid message and substitute someone else's credentials thereby impersonating the other party, or perhaps the attacker is trying to modify an existing message to the behavior of the service. Either way, when a message is digitally signed, the service can verify the message origin and reject the message if its origin is deemed invalid. | Service Composability |
| Direct Authentication | As service capabilities require that consumers provide credentials that can be authenticated against an identity store, the service implementation is provided access to an identity store, allowing it to authenticate the consumer directly. | Direct Authentication: By having the service authenticate consumer requests against an identity store, only safe consumers can access sensitive data and logic. | Service Composability |

## Design Patterns in Cloud Service & Storage Security

**Table 1‑4. Selected Design Patterns in Cloud Service & Storage Security‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Trusted Platform  BIOS |  | | Using security validation from the silicon up and remote monitoring of the platform security status, cloud consumers can verify that they are using trusted platform modules (TPM) that meet their security assurance requirements. | Trusted Platform BIOS: A platform startup sequence with a trusted BIOS. | Digital Sig., Hardware Sec. Module, Trusted Platform Module |
| Geotagging | When trusted resource pools are generated, the geolocation is supplied as part of the compliance and regulatory assurance attributes, using TPMs. | Geotagging: The geolocation is configured in the TPM. | Platform Trust Policy,  Trusted Platform Module. |
| Hypervisor Protection | A system is established whereby mitigations are implemented from the silicon up the stack to protect against hypervisor attack vectors. | Hypervisor Protection: Examples of hypervisor threat mitigations. | Digital Sig., Trusted Platform Module, Host-Based Sec. Sys., Hardened Virtual Server Image. |
| Cloud VM Platform Encryption | Encrypted containers are provided for use and storage of the various types of VM backups and replications. A key manager is used to manage keys for encryption of the various types of VM storage that are pre-provisioned to receive backups and snapshots of consumer VMs or to receive replications and live migrations. | Cloud VM Platform Encryption: Key management and the Cloud VM Platform Encryption pattern. | Cryptographic Key Management System, Virtual Server. |
| Trusted Cloud Resource Pools | Trusted resource pools made up of trusted geotagged computers are made available by the cloud provider, and can be verified by the consumer through direct monitoring or evidence through auditing. | Trusted Cloud Resource Pools: Resources pooled by security assurance level. |  |
| Secure Cloud Interfaces and APIs | A cloud identity and access management (IAM) system is instituted to differentiate intruders from legitimate consumers. An authentication gateway service (AGS) and an IAM system are implemented to identify and authenticate legitimate consumers and grant them access, while denying access to intruders. | Secure Cloud Interfaces and APIs: The AGS authenticates cloud consumers for access to APIs. | Authentication Gateway Service, Identify and Access Management. |
| Cloud Resource Access Control | A cloud single sign-on (SSO) architecture is established, incorporating an authentication gateway service (AGS) and attribute authority for implementation of cloud resource access control. An AGS and attribute authority connected to a secure token service (STS) are implemented and provisioned with the organization’s cloud consumers’ accounts and attributes. | Cloud Resource Access Control: Cloud Consumer A is directed to the AGS for authentication (1). The AGS authenticates Cloud Consumer A and requests an attribute token from Organization A’s attribute service (2). Incorporating an STS function, the attribute service issues an attribute token to the AGS for Cloud Consumer A. The attribute token has a short lifespan, typically around 2 to 3 minutes (3). The AGS provides the attribute token to Cloud Consumer A, which supports access control as an SSO model with attributes of the cloud consumer. The attributes will be used by the cloud service providers to determine access privileges (4). Cloud Consumer A provides the attribute token to Service A to prove authentication and support an access decision for its resources. Cloud Consumer A is also required to use their certificate and a holder-of-key (HOK) assertion as part of the attribute exchange protocol with the cloud service. Without an HOK check, the attribute token can be stolen and used by an attacker (5). Using an SSO architecture, the cloud consumer provides the attribute token to Service B to prove authentication and support an access decision for its resources. Again, Cloud Consumer A is required to do an HOK check as part of the attribute exchange protocol (6). | Security Token Service, Single Sign-On, Cloud-based Security Groups, Attribute Authority, Attribute-based Access Control. |
| Mobile Bring-Your-Own-Device (BYOD) Security | According to the BYOD security requirements and policy developed, a security framework is established to monitor, control and protect the mobile devices, their applications and data, and their network connections. An enterprise mobility management (EMM) system is implemented, covering the special security requirements of mobile devices. | Mobile BYOD Security: Solutions for monitoring, controlling, and protecting mobile devices include the device (A), network (B), and integrating with corporate SIEM and logging systems for visibility and monitoring for compliance (C). | Enterprise Mobility Management System |
| Cloud Data Breach Protection | A system is established that provides encryption of sensitive data so that if it is lost, it is not readable by an attacker. Using validated encryption and governance that meets enterprise security assurance levels, data at rest is encrypted. | Cloud Data Breach Protection: The use of enterprise encryption for sensitive data. | Cryptographic Key Management System, Threat Intelligence System. |
| Permanent Data Loss Protection | A data retention policy is created and implemented as part of a security policy to mandate that backups of data and keys are created and maintained. Data is inventoried and assigned a sensitivity factor and a risk of loss factor, is encrypted and backed up in a location separate from the primary infrastructure, and the cloud consumer verifies that the cloud provider meets the data retention policy. | Permanent Data Loss Protection: Data retention policy, which normally meets a compliance requirement, requires an inventory of all sensitive data, the sensitivity level of the data, the risk of loss and backup schedules and forms (A). Backups of data and encryption keys are created according to policy, and backup copies are stored off-site and separate from the operational data so that risks of loss are mitigated (B). Periodic audits ensure compliance with policy (C). | Cryptographic Key Management System |
| In-Transit Cloud Data Encryption | An encryption mechanism is implemented to encrypt data between sender and receiver for confidentiality, and a digital signature mechanism is implemented to provide integrity for the data. | In-Transit Cloud Data Encryption: An attacker attempts to intercept data uploading into a cloud environment, however, the data is encrypted and signed before it is sent. | Digital Signature, Encryption. CKMS, Cloud Storage Device. |
|  |  |  |  |

## Design Patterns in Cloud Computing – Sharing, Scaling and Elasticity

**Table 1‑5. Selected Design Patterns in Cloud Computing – Sharing, Scaling and Elasticity‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Shared Resources |  | | Physical IT resources are shared by partitioning them into lower capacity virtual IT resources that are provisioned to multiple cloud consumers. Virtualization technology is used to create virtual instances of physical IT resources. Each virtualized IT resource can be assigned to a cloud consumer, while the underlying physical IT resource is shared. | Shared Resources: Each cloud consumer is allocated a virtual server instance of a single underlying physical server. In this case, the physical server is likely greater than if each cloud consumer were given its own physical server. However, the cost of one high-capacity physical server is lower than four medium-capacity physical servers and its processing potential will be utilized to a greater extent. | Audit Monitor, Cloud Storage Device, Cloud Usage Monitor,Hypervisor. |
| Dynamic Scalability | The IT resource can be integrated with a reactive cloud architecture capable of automatically scaling it horizontally or vertically in response to fluctuating demand. | Dynamic Scalability: An example of a dynamic scaling architecture involving an automated scaling mechanism (Part III). | Automated Scaling Listener, Cloud Storage Device, Cloud Usage Monitor,Hypervisor. |
| Service State Management | The cloud service is designed to integrate with a state management system allowing it to defer state data at runtime when necessary so as to minimize its IT resource consumption. A state management system requires a cloud storage device capable of temporarily holding and releasing state data exchanged by the cloud service. | Service State Management: A sample scenario in which cloud service state data is manually deferred and restored. | State Management Database, Cloud Storage Device, Hypervisor and etc. |
| Memory Over-Committing | Memory virtualization is implemented to enable more virtual servers to be hosted on the same host, by allowing the host’s physical memory to be exceeded by the total memory configuration of the virtual servers. | Memory Over-Committing: Virtual Servers A, B, and C are each allotted a portion of Host A’s physical memory. | Hypervisor, Virtual Infrastructure Manager, Virtual RAM and etc. |
| Broad Access | Runtime mapping logic is incorporated to transform data exchanges between the cloud service and different cloud service consumers. The mapping logic is added to the cloud service logic or positioned separately, within a multi-device broker mechanism that transforms data exchanged between the cloud service consumer and the cloud service. | Broad Access: The cloud consumer (top) accesses and configures a physical server using a standard device and protocol that is now supported as a result of applying the Broad Access pattern. The cloud consumer (bottom) later accesses the cloud environment again to install a virtual server on the same physical server, and deploys an operating system and a database server. Both actions represent management tasks that can be accomplished via different devices brokered by the same centralized multi-device broker mechanism. | Multi- Device Broker |
| Command and Query Responsibility Segregation (CQRS) | Segregate operations that read data from operations that update data by using separate interfaces to maximize performance, scalability, and security. | Figure 3 - A CQRS architecture with separate read and write stores | Scalability, Responsiveness and Performance. |
| Event Sourcing Pattern | The pattern defines an approach to handling operations on data that is driven by a sequence of events, each of which is recorded in an append-only store, which can simplify tasks in complex domains by avoiding the requirement to synchronize the data model and the business domain; improve performance, scalability, and responsiveness; provide consistency for transactional data; and maintain full audit trails and history that may enable compensating actions. | Figure 1 - An overview and example of the Event Sourcing pattern | Scalability and Performance. |
| Cache-Aside Pattern | Applications use a cache to optimize repeated access to information held in a data store.  The caching systems provide read-through and write-through/write-behind operations. In this pattern, an application retrieves data by referencing the cache. If the data is not in the cache, it is transparently retrieved from the data store and added to the cache. Any modifications to data held in the cache are automatically written back to the data store as well. | Figure 1 - Using the Cache-Aside pattern to store data in the cache | Scalability and Performance. Data Management. |

## Design Patterns in Cloud Access Management

**Table 1‑6. Selected Design Patterns in Cloud Access Management‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Cloud Denial of Service Protection |  | | A cloud DoS protection service is incorporated into the security architecture to shield the cloud provider from DoS attacks, through updating the domain name service (DNS) to route all cloud provider traffic through the protection service, which filters attack traffic and routes only legitimate traffic to the cloud provider. | Cloud Denial-of-Service Protection: The DDoS mitigation service in action. | Domain Name Service, Traffic Filter, Traffic Monitor |
| Federated Cloud Authentication | Federated authentication allows cloud services to authenticate cloud consumers incorporating a certificate status checking responder on the local area network (LAN), which supports authentication independently when the CRL for a given organization is unavailable due to failure. A local certificate validation service (CVS) is implemented to check the revocation status of submitted certificates from multiple organizations. | Federated Cloud Authentication: The component as a cloud service consumer sends the required certificate to an authentication service to be authenticated (1). The authentication service sends a request to the CVS that includes the issuer and serial number of the certificate (2). The CVS compares the serial number with the associated issuer’s CRL to determine if the certificate is revoked (3). The CVS signs a response indicating if the certificate is good, revoked, or unknown (4). The CVS also checks to see if the CRL is valid. If the CRL is stale and the CVS cannot retrieve a current one, the CVS can be configured to send an alert to a cloud resource administrator (5). | Certificate Validation Service, Certificate Revocation List and etc. |
| Cloud Key Management | A cloud key management system is employed, optionally using a hardware security module (HSM) for key protection, consisting of systems, personnel and policies is implemented to manage keys for encryption of all required data for both on-premise and cloud resources, available either as a physical or virtual network attached device. | Cloud Key Management: A sample CKMS architecture. | Cryptographic Key Management System, Hardware Security Module. |
| Secure Cloud Interfaces & APIs | An authentication gateway service (AGS) and an IAM system are implemented to identify and authenticate legitimate consumers and grant them access, while denying access to intruders. | Secure Cloud Interfaces and APIs: The AGS authenticates cloud consumers for access to APIs. | AGS – Authentication Gateway Service, IAM-Identity Access Management |

## Architecture Patterns ( Supplemental Listing )

**Table 1‑7. Architecture Patterns - Supplementary‑**

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Single Page  Web App |  | | A service consumer invokes a query on a data-providing service (service provider) and then is free to perform whatever tasks it wishes, independent of the service request. |  | Observer Pattern, MVC. |
| Dependency Injection | Implements inversion of control for resolving dependencies. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern. |  | Cross-Cutting Concerns, Inversion of control. |
| Micro-Service Anti-Pattern | Each service to implement end-user authentication, throttle, orchestrate, transform, and route etc. in each service. Some services are built to satisfy one consumer non-functional requirements, but not another’s. |  | Fine Grained Entity Endpoint, Service Composability. |
| Micro-Service Regular Pattern | Invest in API Management solutions to centralize, manage and monitor some of the non-functional concerns and which would also eliminate the burden of consumer's managing several micro-services configurations. API gateway can be used to orchestrate the cross-functional micro-services that may reduce round trips for web applications |  | Fine Grained Entity Endpoint, Service Composability, Cross-Cutting Concerns. |
| Universal Application Container |  |  | Cross-Cutting Features, Shared Common Services, Dependence Injection, Standardization of API. |
| Service Orchestration | Multiple services and APIs are centrally delegated and coordinated through a service hub. |  | Service Composability, Service Loose Coupling |
| Service Choreography | Multiple services, APIs and subservices are coordinated in a decentralized style. |  | Service Composability, Service Loose Coupling |
| Service Composition in Various Granularities and Hierarchies | Tiered services and APIs are composed together with various granularities in order to publish newly formed App & API. |  | Service Composability,  Service Loose Coupling |
| Uniform REST Contract with CRUD | Example of standardized uniform contract in HTTP/REST across alternative service endpoints that is abstracted from the specific capabilities of individual service. |  | Standardized Service Contract, Service Abstraction, Reusability & Composability |
| Attribute-Based Entity-Identity Modeling | A given identity object consists of a finite set of attribute values. These properties record information about the object, either for purposes external to the model or to operate the model, for example in classification and retrieval.  User access enables users to assume a specific digital identity across applications, which enables access controls to be assigned and evaluated against this identity. |  | Entity-Identity Mapping with attributes and identifiers. |
| Wrapper | Encapsulate a legacy service API inside a generic, stateless service. Wrap the interoperation mechanisms within a service façade that operates with the legacy system as if it were a legacy consumer, and exposes a normalized SOA interface to new consumers. | Wrapper | Service Loose Coupling, Standardized Service Contract |

## Services and Systems (Infrastructure) Patterns

Service and Systems patterns allow CA/CST to unravel design issues rapidly and leverage demonstrated solutions. Patterns are guidance that provides CA/CST with techniques to address challenges and issues that regularly transpire within their enterprise. These patterns offer CA/CST ecosystems consistency, repeatability, and structure. Furthermore, since the patterns are field tested, CA/CST is able to exploit a higher quality of design throughout the CA/CST infrastructure.

Table 1‑8. Services and Systems (Infrastructure) Patterns‑

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | Asynchronous Queuing |  | | A service consumer invokes a query on a data-providing service (service provider) and is then free to perform whatever tasks it wishes, independent of the service request. | http://4.bp.blogspot.com/-5Ou09ISyMuU/Un_Yf6cBtOI/AAAAAAAAAP8/iru4LFGRyRI/s1600/async-service-invocation-queuing.png | SP1 |
| Dynamic Scalability, Failure Detection and Recovery | Dynamic horizontal scaling is enabled via the use of pools of identical IT resources and components capable of handling fluctuating workloads across each pool. Dynamic vertical scaling increases the processing capacity of a single IT resource. Public cloud vendors commonly provide horizontal scaling capabilities that are configurable based on a variety of metrics such as CPU utilization, data transfer rate etc. Different intelligent monitoring and recovery technologies can be used to establish the automation of failure detection and recovery tasks. Such technologies focus on watching, deciding upon, acting upon, reporting, and escalating IT resource failure conditions. |  | SP6 |
| Failover/Recovery | In order to ensure business continuity, applications and their underlying infrastructure must be designed to support restoration of services within appropriate recovery point objectives.  Depending on criticality of service, consider either active/passive or active/active DR setup. In an active/passive setup, application configuration and data is restored from off-site back The resulting latency in service restoration is usually acceptable to system with low-criticality. For systems with higher criticality, active/active failover needs to be considered where primary and secondary hosting sites are always in sync through mechanisms such as replication. The only delay in recovery is in changing DNS tables and other routing mechanisms for redirecting live traffic to the alternate site. |  | SP6 |
| Geographic Load Balancing | A global load balancer is used to dynamically route requests by calculating virtual server instances and associated workloads and correspondingly distribute the processing across available physical server hosts. |  | SP6, SP5 |
| Publish and Subscribe Interface | Publisher service broadcasts data by different topics to those that subscribe to the particular topic area. | https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcRzS4cqoTcv4pj_lcdVDYkPXjfswKN-5yRoXDc74xsz7f8G92rIOg | SP1, SP6, SP7 |
| Legacy Wrapper Services and Systems (Infrastructure) Patterns | A wrapper service is tightly coupled to a legacy artifact. The interface the legacy wrapper provides is standardized, effectively decoupling consumers from the legacy wrapper implementation as well as the legacy artifact implementation. | http://cdn.ttgtmedia.com/digitalguide/images/Misc/pl_soa_fig6.gif | SP3, SP7 |
| Lightweight Service | A service that exposes the same interface as the full service but initializes only the needed operations. |  | SP1, OP2 |
| |  |  | | --- | --- | | Stateless Service |  | | Each call to the service passes on all of the information the service needs to fulfill the request. This allows services to scale through deploying more instances of the service that operate independently of the other instances. |  | SP2, OP2 |
| |  |  | | --- | --- | | Stateful Service |  | | A stateful utility service is used as an intermediary to provide in-memory state data storage and/or to maintain service activity context data. |  | SP4 |
| Service Virtualization | Service virtualization a pattern, which applies abstraction of a concrete service through a proxy or intermediary service. With this pattern, the service proxy interfaces with the service consumer (i.e. a service or application initiating a request) and encapsulates the physical characteristics of the service provider (i.e. a service or application providing functionality). |  |  |
| Service Prioritization | With service selection pattern, several instances of a service provider with the equivalent interface are exposed to the service consumer at a same endpoint address. Each instance of the service provider provides a different level of service and the service proxy routes a request to the appropriate endpoint based on the context of the service consumer’s invocation or service policy. It is the service proxy’s responsibility to choose the suitable version based on the service consumer’s needs or service policy. |  |  |
| Service Mediation | The service mediation pattern is when a service provider is accessible through a service proxy with a varying interface than the original interface. This enables limited functionalities to be exposed while meeting the requirements of a service consumer. The service proxy then mediates the consumer request to meet the requirements of the original interface to routing it to the service provider. |  |  |
| Extract, Transform, Load (ETL) | Extract, Transform and Load (**ETL**) is the process of extracting data from homogeneous or heterogeneous data sources; transforming it for storing it in proper format or structure for querying and analysis purpose. | https://www.passionned.com/wp/wp-content/uploads/data-warehouse-and-etl-what-is-etl.png | DP1, DP3 |
| Business Intelligence (BI) | BI is an architectural pattern that encompasses a variety of tools, applications and methodologies which enable organizations to collect data from internal systems and external sources, prepare it for analysis, develop and run queries against the data, and create reports, dashboards and data visualizations to make the analytical results available to corporate decision makers as well as operational workers. BI data is typically stored in Data Warehouses or smaller Data Marts that hold smaller subsets of enterprise data. |  | DP3, DP2, DP1, OP3 |
| Service Oriented Architecture (SOA) | SOA is a pattern of building software from discoverable, loosely coupled, reusable services via standard communication protocol over a network. | http://www.w3.org/2003/Talks/0521-hh-wsa/soa.png | OP3, OP5, SP2, SP4, SP7 |
| Enterprise Service Bus (ESB) | An ESB is an architectural pattern that decouples systems from each other, allowing them to communicate without dependency on or knowledge of other systems on the bus. It supports moving away from point-to-point integration, which becomes brittle and hard to manage over time. Point-to-point integration results in custom integration code being spread among applications with no central way to monitor or troubleshoot. | https://research.linagora.com/download/attachments/3050118/esb.png?version=1&modificationDate=1297354481000 | SP7, SP4, SP5, SP3, SP1 |
| Enterprise Resource Planning (ERP) | ERP is an application pattern that helps manage core enterprise information and processes such as supply chain, procurement, inventory, finance, product lifecycle, projects, human resources and other mission-critical processes. | http://www.greenbeacon.com/GreenBeaconWebsite/images/default-source/inline-images-content/ERP_2.jpg?sfvrsn=0 | OP3, SP1, DP1 |
| Business Process Management (BPM) | BPM is an approach to improving an organization’s business processes. It fosters ongoing collaboration between IT and business users to jointly build applications that effectively integrate people, process and information.  BPM gives an organization the ability to define, execute, manage and refine processes that  -involve human interaction, such as placing orders  -work with multiple applications  -handle dynamic process rules and changes | http://www.slginnovation.com/sites/default/files/imagefield_thumbs/bpm-flowchart.png | OP1, OP5, SP1, SP2, SP4, SP5 |
| Customer relationship management (CRM) | CRM is a software application pattern that helps manage interactions with customers, prospective customers and information supporting sales activity as well as post-sales activities such as customer support. | http://www.superoffice.com/blog/wp-content/uploads/2013/11/crm-strategy.png | OP3, OP1, DP1, SP2, |
| Web Portal | A web-portal is a generic self-service web application that organizes information and system functionality to facilitate ease of access to the user. |  | OP2, DP3, SP3, SP2, SP1 |
| Content Management System (CMS) | A CMS enables users to self-publish their content over the web in a variety of formats. Content may include text, multimedia files or any other files type that follow the content management lifecycle. |  | OP2, OP3, DP1, DP, SP2 |
| Batch Processing | Certain information exchanges with partners are high volume and can therefore only be file based. In such cases batch processing methods may still be applicable as a realtime data exchange and processing may not be feasible. An example of this in CST environment is the Citibank Lock-Box process for the injest of transcribed passport applications. |  |  |

## Data Patterns

The following data patterns address issues that are commonly encountered across the CA/CST enterprise. The objective of data patterns is to directly the complexity introduced by data management systems, and provide solutions to common problems. These patterns are relevant to the DIV.

Table 1‑9. Data Patterns‑

| *Pattern* | *Description* | *Diagram* | *Mapped Principles* |
| --- | --- | --- | --- |
| Transactional  Data Store | Systems such On-Line Transaction Processing (OLTP) Systems, Enterprise Resource Management Systems (ERPs), and other “back-office” systems follow a persistence pattern where they use databases based on RDBMS technology which is generally optimized for transactional uses. These databases provide ACID guarantees that are required by business transactions. |  | DP3 |
| Analytical Data Store | Historical data about business metrics such as consular workforce performance, visa/passport application processing times and such are typically stored in read-only repositories that are optimized for high performance queries and reports. This information is updated from transaacitonal databases on a regular basis. Typcially used by BI and Data Warehouses, there are 5 types of analytic databases namely Columnar, In-memory, Massively Parallel Proessing databases, Datawarehousing applicances and OLAP databases. |  | DP1,DP3 |
| Content Store | A content repository is a hierarchical content store with support for structured and unstructured content, full text search, versioning, transactions and more. A content repostiory acts as a persistence store for a Content Management System or a Document Management System. These systems add a user interface on top of the API exposed by repository. |  | DP1, DP3 |
| Distributed Data Stores | Distributed Databases provide high levels availability by spreading data across multiple nodes on a cluster. These databases typically operate with non-relational formats such as JSON, XML and BSON. They also do not use SQL for data manipulation and are therefore generally referred to as NoSQL DBs. These databases are increasingly used in big data and real-time web applications. |  | DP3 |
| Data Caching | Repetitious acquisition, initialization, and release of the same resource causes unnecessary performance overhead. In situations when the same component or multiple components of a system  access the same resource, repetitious acquisition and initialization incurs cost in terms of CPU cycles and overall system performance.  Temporarily store the resource in a (cheap) buffer called a cache. Subsequently, when the resource is to be accessed again, use the cache to fetch and return the resource instead of acquiring it again from the resource environment such as an operating system that is hosting resources. The Cache identifies resources by their identity, such as pointer, reference, or primary key. Commercial and open source implementations of in-memory grids provided thie data caching functionality to applications. |  | DP3 |
| Ensuring Data Consistency | CST applications typically use data that is dispersed across multiple data stores across the enterprise. Managing and maintaining data consistency in such an environment is a critical aspect of the system. Due to the distributed-ness of data stores that are typically part of a transaction at CST, solutions must be designed for eventual consistency accepting that the data across other applications might not be consistent all the time. |  | DP1 |
| Master-Data Quality, Consistency and Accuracy | Consistency, quality and accuracy of master data are critical to the efficient and error-free execution of business processes and to produce accurate information for decision makers. Transactional systems have a huge dependency on master data for operational integrity and accuracy. Implement a Master Data Management (MDM) solution to identity master data, its sources, and governance processes for its management and retrieval by various transactional systems. |  | DP1 |
| Analysis of Large Datasets | To satisfy queries or to generate reports that require analyzing large and diverse data sets in a timely manner, conventional data processing applications & RDBMSs are indeqaute. A distributed parallel architecture spreads the data across multiple processing units, which provide data much faster and improve the overall processing speed. Data is inserted into a parallel DBMS, which implements Google’s MapReduce and Apache Hadoop frameworks. The processing power of the system is made transparent to the end user who only communicate with a front-end web/application server for query or report generation. |  | DP3 |

## Information Assurance (IA) Patterns

This section describes patterns relevant to the IA viewpoint. The IA patterns document ways to accomplish IA functions such as authentication, authorization, and secure access to data in a services environment. These architectural building blocks include security services and physical elements that provide security. The implementation of the patterns may introduce variations to accommodate the security products used and the risk tolerance of the operational environment. Nevertheless, the patterns described here present general approaches that should be considered for programs at CA. The graphic in the IAV (Section 2.6), depicts a notional architecture of how these patterns can be implemented. Table 5-3 lists patterns that should be considered for use.

Table 1‑10. Information Assurance Patterns‑

| ***Name*** | ***Description*** | ***Diagram*** | ***Mapped Principles*** |
| --- | --- | --- | --- |
| Transport-Level Security | Provides point-to-point transport confidentiality of a message between the service client and service provider. This is used when traversing security enclaves. |  | SECP2 |
| Client-authenticated TLS using PKI | Service client authenticates directly with a service provider using public key-based, mutual TLS. Alternatively, an authentication gateway in the service provider’s domain may be situated in-line between the service client and the service provider to offload authentication functions from the service provider. |  | SECP2 |
| ABAC Authorization | Atribute-based access control (ABAC) is a security pattern in which access rights to a protected resource are granted through the evaluation of user attributes. There are 2 ways attributes are made available to a Polic Enforcement Point (PEP.) 1- The service consumer passes attributes to the service provider for an access control decision. 2- A Policy Decision Point (PDP) retrieves all attributes about the service consumer to make an access control decision. |  | SECP2 |
| Message-Level Digital Signature | Digital signatures provide service clients a mechanism of proving their identity to the service provider. Aditionally they also garauntee message integrity during transit to the service provider. The clients signs the message with its private key which the provider verifies using the clients public key. |  | SECP2 |
| Message-Level Encryption | Message-Level Ecryption allows a client to encrypt parts of its message to the provider to ensure confidentiality of either a part or the complete message. This can be used in conjunction with Message-Level Signature to provide end to end security. |  | SECP2 |
| SOA Gateway/ XML Firewall | A SOA Gateway is an appliance that specializes in Web Service security. These typically support the breadth of security standards to enable secure consumption and exposure of enterprise services in interoperable manner. |  | SECP1  SECP2 |
| Identity and Access Management (IAM) | An IAM is a security pattern where common security services such as web single-sign on, identity and access provisioning, identity lifecycle management, and federated access control, are made available through a COTS products |  | SECP1  SECP2 |
| Managing large number of certificates | A Managed PKI service provides centralized management of certificates needed across the enterprise be it for online transactions with customers, web service/api based process automation, securing firewall/VPNs etc. Functions typically provided under this service are, automated renewal of certificate, delegated administration, certificate revocation list service, online certificate status protocol service and web-based enrollment and provisioning. |  | SECP1  SECP2 |
| Disparate Identity Stores | A virtual directory or virtual directory server is a software layer that delivers a single access point for identity management applications and service platforms. A virtual directory operates as a high-performance, lightweight abstraction layer that resides between client applications and disparate types of identity-data repositories, such as proprietary and standard directories (Active Directory), databases (CST Tables, PPT tables), web services, and applications. |  | SECP1  SECP2 |
| Protecting sensitive resources with tiered networking | The general philosophy behind a tiered network architecture is that the front-end (less trusted) systems could be compromised, yet the network is designed with enough tiers to allow administrators to detect an attack and take action before an adversary can gain access to a back-end system (most trusted). Any systems accessible from the internet must be in the DMZ and shouldn’t contain sensitive data. Sensitive data should be in a private network. DMZ systems must go through an application server to get to the system on the private network. |  | SECP1  SECP2 |
| Securing data in use | Data in use, or memory, can contain sensitive data including digital certificates, encryption keys, intellectual property (software algorithms, design data), and personally identifiable information. Compromising data in use can enables unauthorized access to encrypted data at rest and data in motion. Data in use can be secured by techniques such as full-memory encryption, CPU-based keystorage and through use of special cryptographic protocols such as secure multi-party computation and homomorphic encryption. |  | SECP1  SECP2 |
| Securing data at rest | Data inside of files of DBMS and file servers is vulunerable to unauthorized access when it is resides as plain text or is unencrypted. Data at rest should be encrypted with strong encryption methods such as AES, RSA, and SHA-256. Increasing encryption on multiple levels is recommended. Cryptography can be implemented on the database housing the data and on the physical storage that the databases utilize. Data encryption keys should be updated on a regular basis. Encryption keys should be stored separately from the data. Periodic auditing of sensitive data should be part of policy and should occur on scheduled occurrences. Additionally, only store the minimum amount of sensitive data. |  | SECP1  SECP2 |

# Reference Architecture Principles & Rules

## CA Architecture Principles and Associated Rules

These architectural principles are guidelines for development and implementation of all CA IT solutions, resources, and assets. They reflect a level of consensus across the enterprise, and form the basis for which future IT decisions are made.

Table 4-2 contins a set of Operational Principles which are more general in nature, whereas the subsequent principles are more specific, addressing the topics of Data, Service Oriented Architecture (systems/services), and Security (IA). The table below not only lists the principles in the manner defined above, but also provides insight into the rationale and technical implications of each architectural principle. Touchpoints between the CA and CST are highlighted, and provide insight into how the architecture supports the business/mission. The last column provides the rule(s) that align to each principle.

Table 2‑1-APR. CA Architecture Principles and Rules‑

| ***Operational Principles*** | | | | |
| --- | --- | --- | --- | --- |
|  | ***Principle*** | ***Rationale*** | ***Implication(s)*** | ***Rules*** |
| **OP1** | Solutions are architected to achieve business goals and advance bureau strategies | To ensure operational and financial efficiency, it is imperative that the IT initiatives and projects are executed in support of business goals, strategies and initiatives. | A strong alignment between IT and business initiatives, reduces the likelihood of funding projects that don’t provide mission value. | * IT project shall align with Bureau strategies and business goals |
| **OP2** | Opt for design implicity. | Simple IT solutions are easier to use, maintain, and modify. | Risk is reduced, processes are more streamelined, and CA operates more efficiently. | * Reduce complexity by reusing components, and leveraging platforms that scale * Solutions will adopt the established user experience design standards * User interfaces will be simple, consistent, intuitive and accessible |
| **OP3** | Adopt, buy before create | Serves as an enabler to maximize component reusability and platform scaleability through the application of service-oriented architecture and the adoption of standards | CA can better deploy IT services to strategically transform its business.  CA can efficiently scale their opperations and leverage different providers as performance needs change. | * Solutions will consider service-based approaches before other alternatives |
| **OP4** | Architectural impacts must be assessed throughout the lifecycle, beginning at the conceptual phase | Assessing architectural impacts not only enables the enterprise to better understand the proposed solution, but also ensures the fit of the solution into the existing environment.  Identifying and managing risks early in the solution development lifecycle tremendously reduces the chances of missing a solution, schedule, or resource target. | CA will realize a higher rate of solution implementation success. New solutions will be rolled out, on time and within budget. New solutions will meet interoperability requirements with existing systems. | * Developers must follow CST lifecycle governance processes * Compliance with CA EA is a prerequisite for IT investments |
| **OP5** | Apply Federal, Departmental, Bureau, commercial and open\* standards[[1]](#footnote-1) with consistency. | CA’s systems not only provides data and services in support of worldwide Consular operations, but also supports a variety of external agency information needs. Standards are vital as they facilitate change amongst community (CA and external agency) systems, and improve their level of interoperability.  Standards promote the definition of reusable services, and foster secure data exchange across the government. | The solution development and delivery time is reduced as a result of the application of standards.  CA EA team must partner with the CA technology stakeholders to identify which emerging standards are applicable to the enterprise.  The CA EA program must capture, assess, and report the impact of each standard on the performance architecture. | * Solutions must adhere to Federal, Departmental, and Bureau, commercial and open standards as applicable |

Table 2‑1-DSP. Data Specific Principles‑

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Data-Specific Principles*** | | | | |
|  | **Principle** | **Rationale** | **Implication(s)** | **Rules** |
| **DP1** | Data is a shared enterprise asset – data definitions and vocabularies will be consistent throughout the organization. | In order to realize a person-centric architecture data definitions need to be consistent across the enterprise. | A data governance, data management and data quality processes must be incorporated into the existing governance process. | * Data will be validated at the point of collection |
| **DP2** | Data is an asset that must be protected | CA captures citizens personal information and must protected to maintain citizens confidence in CA services. | As part of the enterprise security architecture that is “baked into” the enterprise rather than layered on as an afterthought the protection of PII is critical. | * Solutions shall include a data preservation strategy * Sensitive data shall be protected (in use, in transit and at rest) * Solutions shall provide the capability to support data choreography, to address potential “data collisions” |
| **DP3** | Users will enter their data once and be used as required throughout the enterprise | A person-centric vision of the future streamlines the process by which services are provided. | The user experience will be a major driver to the architecture and design of the future systems/services. | * Solutions will be designed to reuse exising data and minize the amount of information requested from return customers |

Table 2‑1-SIP. System/Service Interoperability Principles‑

| *System/Service Interoperability Principles* | | | | |
| --- | --- | --- | --- | --- |
|  | Principle | Rationale | Implication(s) | Rules |
| SP1 | Separation of concerns – the architecture is designed to minimize the impact of changing components. | The minimization of interaction points within and between systems achieve high cohesion and low coupling. | Minimizes architectural debt while maximizing flexibility to maintain capabilities. Futhermore, increases reusablitiy of capabilities and increases adaptability to environment change. | * Solutions shall be developed to integrate into a layered/tiered architecture |
| SP2 | Extensibility – applications will be designed to accommodate future growth | Reduce impact of adding new features on current code, data and functionality. Reusable components are generally extensible. | System can be extended with lower cost when adding new capabilities. Increase overall lifespan of systems and applications. | * Applications will provide designs that allow functionality to be extended in the future |
| SP3 | Reusability – applications/system components will be modularly architected for reuse/composition with other systems | Reuse across systems will reduce development time and increase commonality/consistency across solutions | Software components need to be designed so they can be built, packaged, distributed, installed, configured and upgraded independently | * Application/system components will be architected to be reusable and modular |
| SP4 | Monitoring and measurement – applications will be designed to support monitoring and measurement | Systems must support monitoring for performance and outages to ensure continued availability and adherence to SLAs. | Systems must be able to leverage enterprise tools for monitoring and management. Alternatvely they should provide tools and methods for the same. | * Application/Service will support monitoring using existing tools where possible |
| SP5 | Clearly published SLAs- Applications will adhere to and publish SLA agreed upon with business. | Applications service levels must be understood/agreed upon by business and must align with the KPIs established by business. | Application SLAs dictate the design of infrastructure elements such as network, compute and storage. | * Application will have clearly defined SLA agreed upon and understood by business * To continually meet SLA and future business requrements, capacity management will be incorporated into solutions |
| SP6 | High availability – applications will be architected for high availability to provide continuity of business operations. | Aplication will be architected for appropriate availability targets based upon the mission criticality of the system. | Applications must be architected to support HA techniques including but not limited to load balancing and clustering. | * Applications will be architected to scale horizontally when allocated more resources * Applications will handle errors in a controlled fashion and continue to operate normally providing graceful degradation. |
| SP7 | Services will be loosely coupled and abstract. | Loosely coupled components have a small amount of direct knowledge of each other. This proves to be beneficial later down the line when service components need to be expanded or features need to be added. | If the project increases in size, each change or update gets increasingly harder the more tightly coupled the components are. Being loosely coupled enables you to keep moving forward with minimal impact to schedule and budget, relatively. | * Solutions will have a standardized interface * Services will be self describing * Service will strive for being stateless * Services will be autonomous – services should have significant control over the functionality (self-healing and self-aware) |

Table 2‑1-SP. Security Principles‑

| ***Security-Specific Principles*** | | | | |
| --- | --- | --- | --- | --- |
|  | **Principle** | **Rationale** | **Implication(s)** | **Rules** |
| **SECP1** | Security should be integrated, and not a discipline performed as an afterthought. | Security should be embedded in the design of the business, application, data and technology architecture layers. | When security is an afterthought, risk is introduced into the deployment of the new capability. | * Security will be a integral part of the development process * Solutions will provide audit trail information * Security requirements – system requirements must specify security features, controls, and operational practices |
| **SECP2** | Employ a defense in depth approach to security. | The CA enterprise employs multiple layers of security controls to ensure redundancy, in the event of a failed security control fails, or an exploited vulnerability in the security architecture. | The layered approach positions CA to be less vulnerable, and well defended against cyber attacks. | * Every layer of the architecture will be secure on its own using detective and preventive controls to defend against vulnerabilities and cyberattacks. * Data at rest shall be encrypted * Any data that crosses security enclaves shall be encrypted in transit * User management – all systems must have defined processes for requesting, issuing, and closing user accounts * Least privilege – Security privileges will be just enough to perform requisite activities based on user roles and attributes. |

# Alphabetical List of Patterns

ABAC Authorization Information Assurance 31

Adapter Classic OOP 4

Analysis of Large Datasets Data 30

Analytical Data Store Data 28

Asynchronous Queuing Infrastructure 20

Attribute-Based Entity-Identity Modeling Architecture 20

Batch Processing Infrastructure 27

Boundary Guard Service Security 7

Broad Access Cloud Computing 14

Brokered Authentication Service Security 8

Business Intelligence (BI) Infrastructure 24

Business Process Management (BPM) Infrastructure 26

Cache-Aside Pattern Cloud Computing 15

Client-authenticated TLS using PKI Information Assurance 31

Cloud Data Breach Protection Cloud Security 12

Cloud Denial of Service Protection Cloud Access 16

Cloud Key Management Cloud Access 16

Cloud Resource Access Control Cloud Security 11

Cloud VM Platform Encryption Cloud Security 10

Command and Query Responsibility Segregation (CQRS) Cloud Computing 14

Content Management System (CMS) Infrastructure 26

Content Negotiation SOA with REST 5

Content Store Data 28

Customer relationship management (CRM) Infrastructure 26

Data Caching Data 29

Data Confidentiality Service Security 8

Data Origin Authentication Service Security 8

Decorator Classic OOP 5

Dependency Injection Architecture 17

Direct Authentication Service Security 9

Disparate Identity Stores Information Assurance 33

Distributed Data Stores Data 28

Dynamic Scalability Cloud Computing 13

Dynamic Scalability, Failure Detection and Recovery Infrastructure 21

Endpoint Redirection SOA with REST 6

Ensuring Data Consistency Data 29

Enterprise Resource Planning (ERP) Infrastructure 25

Enterprise Service Bus (ESB) Infrastructure 25

Entity Linking SOA with REST 5

Event Sourcing Pattern Cloud Computing 14

Exception Shielding Service Security 7

Extract, Transform, Load (ETL) Infrastructure 24

Factory Classic OOP 4

Failover/Recovery Infrastructure 21

Federated Cloud Authentication Cloud Access 16

Geographic Load Balancing Infrastructure 22

Geotagging Cloud Security 9

Hypervisor Protection Cloud Security 10

Idempotent Capability SOA with REST 6

Identity and Access Management (IAM) Information Assurance 33

In-Transit Cloud Data Encryption Cloud Security 12

Iterator Classic OOP 4

Legacy Wrapper Services and Systems Infrastructure 22

Lightweight Entity Endpoint SOA with REST 6

Lightweight Service Infrastructure 22

Managing large number of certificates Information Assurance 33

Master-Data Quality, Consistency and Accuracy Data 29

Memory Over-Committing Cloud Computing 14

Message Screening Service Security 7

Message-Level Digital Signature Information Assurance 32

Message-Level Encryption Information Assurance 32

Micro-Service Anti-Pattern Architecture 18

Micro-Service Regular Pattern Architecture 18

Mobile Bring-Your-Own-Device (BYOD) Security Cloud Security 11

Observer Classic OOP 4

Permanent Data Loss Protection Cloud Security 12

Protecting sensitive resources with tiered networking Information Assurance 34

Publish and Subscribe Interface Infrastructure 22

Reusable Or Uniform Contract SOA with REST 5

Secure Cloud Interfaces & APIs Cloud Access 11

Securing data at rest Inforamtion Assurance 35

Securing data in use Information Assurance 34

Service Choreography Architecture 19

Service Composition in Various Granularities and Hierarchies Architecture 19

Service Mediation Infrastructure 24

Service Orchestration Architecture 19

Service Oriented Architecture (SOA) Infrastructure 25

Service Prioritization Infrastructure 23

Service State Management Cloud Computing 13

Service Virtualization Infrastructure 23

Shared Resources Cloud Security 13

Single Page Web App Architecture 17

Singleton Classic OOP 3

SOA Gateway/ XML Firewall Information Assurance 32

Stateful Service Infrastructure 23

Stateless Service Infrastructure 22

Transactional Data Store Data 27

Transport-Level Security Information Assurance 31

Trusted Cloud Resource Pools Cloud Security 10

Trusted Platform BIOS Cloud Security 9

Trusted Subsystem Service Security 7

Uniform REST Contract with CRUD Architecture 19

Universal Application Container Architecture 18

Web Portal Infrastructure 26

Wrapper Architecture 20

1. Open Standard – format or protocol that is subject to full public assessment and use without constraints in a manner equally available to all parties, managed and further developed independently of any single vendor in a process open to equal participation of competitors and third parties (e.g. IETF, ISO W3C standards) [↑](#footnote-ref-1)