Digital Locker Technology Framework

Version 1.1

Introduction

Currently, in India, many of the documents issued by government or non-government entities are in physical form. This means every time an individual needs to share the document with an agency to avail any service, an attested photo copy either in physical form or on scanned form is shared. Use of physical copies of document creates huge overhead in terms of manual verification, paper storage, manual audits, etc., incurring high cost and inconvenience. This creates problem for various agencies to verify the authenticity of these documents, thus, creating loopholes for usage of fake documents/certificates. In addition, lack of strong identity of the owner that is attached to these documents, it is easy to misuse someone else's document.

Digital India Vision

Digital India aims to transform India into a knowledge based economy. It is an ambitious program and aims to deliver good governance to citizens by synchronized and coordinated engagement with both Central government & State government. Delivery of service through e-Governance represents a paradigm shift and the key is to ensure that the right skills are made available for various stakeholders across the implementation spectrum. This shift requires considerable enhancement of capacities for visualizing, conceiving and delivering projects aimed at transforming existing systems. This requires knowledge of domain as well as technical and techno-commercial-legal capabilities in different levels of government officials. Above all, it requires a basic change in the outlook and functioning of government, so that it becomes citizen-centric rather than process-centric.

The key vision areas under the Digital India Programme are to "provide shareable private space on a public cloud" and to "digitize all documents and records of the citizens and make them available on a real-time basis". This means that "providing citizens with easy access to a shareable private space on a public cloud can greatly facilitate process reengineering through paperless processes. Documents can be issued in verifiable electronic format, made available in various e-Document repositories; citizens can digitally store their documents in any of their preferred digital locker, and then share them with various agencies without the need to physically submit them. This mechanism of 'e-Document repositories' and 'Digital Lockers' will greatly improve the citizen convenience and usher in paperless transactions across the entire ecosystem of public services. All these must be with due user authentication, consent, audits, and other security best practices. This easy access to the digital resources ensures that citizens are not asked to provide government documents or certificates, which are already available with some department/institution of the government, in physical form. Individuals should have an easy way to provider their consent electronically and share various documents when availing a service.

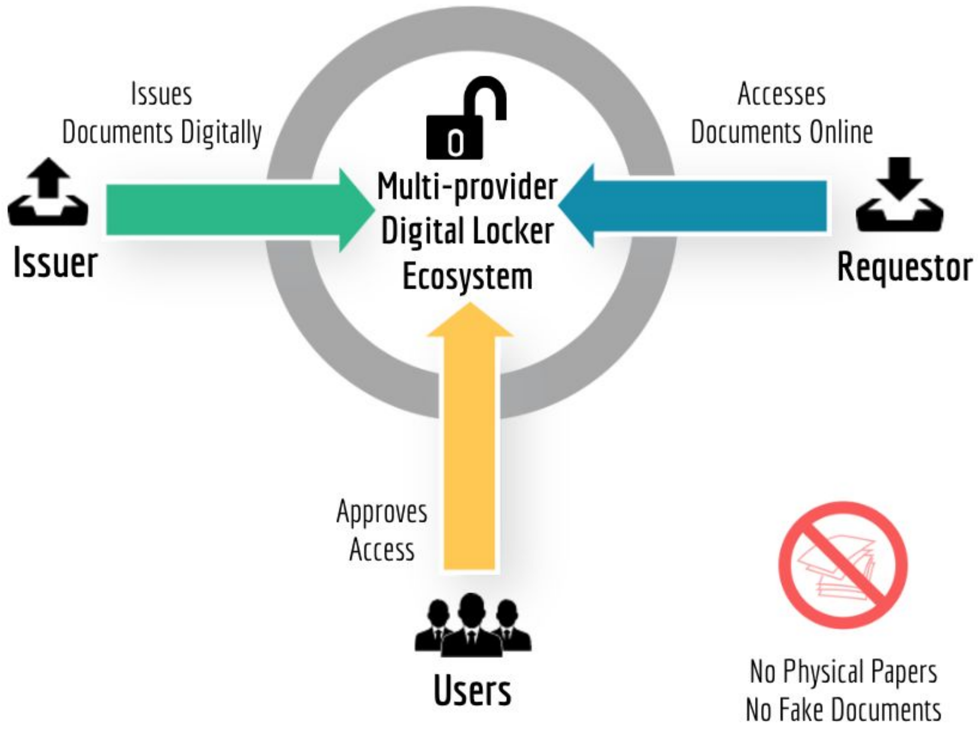
This entire ecosystem of e-Document repositories and Digital Lockers for accessing e-Documents via a standard set of interoperable APIs are together covered under this 'Digital Locker Technology Framework'.

Digital Locker Objectives

I. Enable digital empowerment of individuals by providing them with a choice of Digital Locker on the cloud offered by one of the providers.
II. Enable self signing via e-Sign and make them available electronically.
III. Minimize the use of physical documents.
IV. Ensure authenticity of the e-documents and thereby eliminate usage of fake documents.
V. Secure access to various Government issued documents.
VI. Reduce administrative overhead of Govt. departments and agencies using electronic documents thus making it easier for the individuals to receive services in a paperless manner.
VII. Anytime, anywhere access to their documents by the individuals.
VIII. Open and interoperable architecture for creating a multi-provider ecosystem providing choice to the issuers, requesters, and individuals. This also allows rapid digitalization across various systems.
IX. Architecture to support a well-structured standard document format to support easy sharing of documents across departments and agencies.
X. Ensure privacy and security through user authentication and consented access to documents.

Digital Locker Ecosystem

The following figure depicts the Digital Locker Landscape. Citizens, Issuers, Requestors and Digital Locker are the main components. Digital Locker links various issuer repositories using a set of APIs.



The Digital Locker system has been envisaged as an ecosystem where multiple service providers will provide Digital Locker services such as Locker portals and Repository services for storing of e-Documents. To ensure seamless integration with various Digital Locker service providers as well as other stakeholders, an overarching architecture in the shape of Digital Locker Technical Specifications (DLTS) has evolved and is described in the next section. Open standards based API specifications to standardize open communication between Digital Locker service providers have also been defined and are placed at subsequent sections of this framework.

Digital Locker Technology Specifications (DLTS)

Digital Locker system consists of e-Documents repositories and Digital Lockers for providing an interoperable, federated, and online mechanism for issuers to store and requesters to access various digital documents in real-time with authorized user consent.

Key Terminology

1. ***Electronic Document or E-Document*** – A digitally signed electronic document in PDF or XML format issued to one or more individuals or entities.

Some examples:

- Degree certificate issued to a student by a university.
- Marriage certificate issued to two individuals by a state government department.
- Bank statement of an individual or set of individuals
- Healthcare records of an individual

2. ***Repository*** – A software application complying with specifications under this framework, hosting a collection (database) of e-documents and exposing a standard API for secure real-time access.

3. ***Digital Locker*** – A dedicated secure storage space assigned to an individual or entity, to store e-documents and/or links to e-documents in repositories. The digital locker provider must ensure security (through encryption) and confidentiality (protection against unauthorized access) of these documents and only share with authenticated consent of the user against which the documents are stored. Digital locker providers may provide web/mobile/other application interfaces to the users to manage their profile, security, their documents, view access logs, and manage any consents in addition to providing standard APIs as per this specifications. Digital Locker providers can acquire and sign up issuers and requesters to provide seamless access to e-documents across the entire ecosystem.

4. ***Issuer*** – Any public or private sector entity/organization/department issuing ***digitally signed*** e-documents to individuals/entities and making them available within a repository for access through a digital locker of their choice. The issuer is also responsible to revoke/invalidate their own documents.

5. ***Requester*** – An entity/organization/department requesting secure access with user consent to specific e-documents stored across the ecosystem to provide paperless service to end users.

6. ***Document URI*** – A unique document URI mandatory for every e-document. This unique URI can be resolved to a full URL to access the actual document in appropriate repository.
    - Document URI is a persistent, location independent, repository and issuer independent representation of the ID of the document.
    - The existence of such a URI does not imply availability of the identified resource, but such URIs are required to remain globally unique and persistent, even when the resource ceases to exist or becomes unavailable.

    - While document URI itself is not a secret, access to the actual document is secure and authenticated.

Characteristics of Electronic Documents

To meet the key goals and the solution objectives, architecture should ensure that all electronic documents stored in digital repositories are:

- **Machine Readable** – documents in electronic format should ideally be machine readable (XML, JSON, etc.) instead of formats like PDF eliminating human workflow within requester system. Machine readable e-documents should ideally adhere to common XML structure for application usage and interoperability.
    - Meta Attributes - Documents should have a common set of meta attributes such as issuer ID, document ID, issue date, identity of the owners (individuals/entities) to whom the document is issued to, expiry if any along with document type (domain) specific sub data structure (e.g. school certificates will have different data elements compared to marriage/caste certificates).
- **Printable** – all e-documents should have a printable format attached to it allowing continued printing of certificates for individuals and for backward compatibility with existing paper based systems.
- **Shareable** – users can easily share the documents with other agencies just by providing the unique document URI without having to share photocopies. Such sharing can easily be done even on feature phones even via SMS and text based systems.
- **Tamper Evident** – documents in electronic form must be digitally signed and timestamped by the issuing agency which allows any tampering to be detected. This also allows agencies to be compliant to IT Act.
- **Verifiable** – most importantly, all documents and certificates issued by an issuer can be verified by validating the digital signature of the issuer, thus, eliminating the need for physical verification of the document. In addition, Aadhaar attached documents/certificates ensure only the owner Aadhaar holder can indeed use the certificate, thus eliminating misuse of someone else’s certificates.
- **Secure** – it is critical that documents in the repositories are secure in terms of storage and access. In addition, specific documents (based on

type of document) may only be shareable via owner authentication to ensure sharing and access is authorized by the document owner.

It is highly recommended that government issued documents to individuals have a strong identity such as Aadhaar. When digital documents/certificates are not attached to a strongly verifiable identity, it is important to note that, while those documents can be still made available online in electronic format, it can potentially be misused by another person who has same name/gender etc. It is difficult to verify if the document was issued to same individual without affixing a real identity such as Aadhaar.

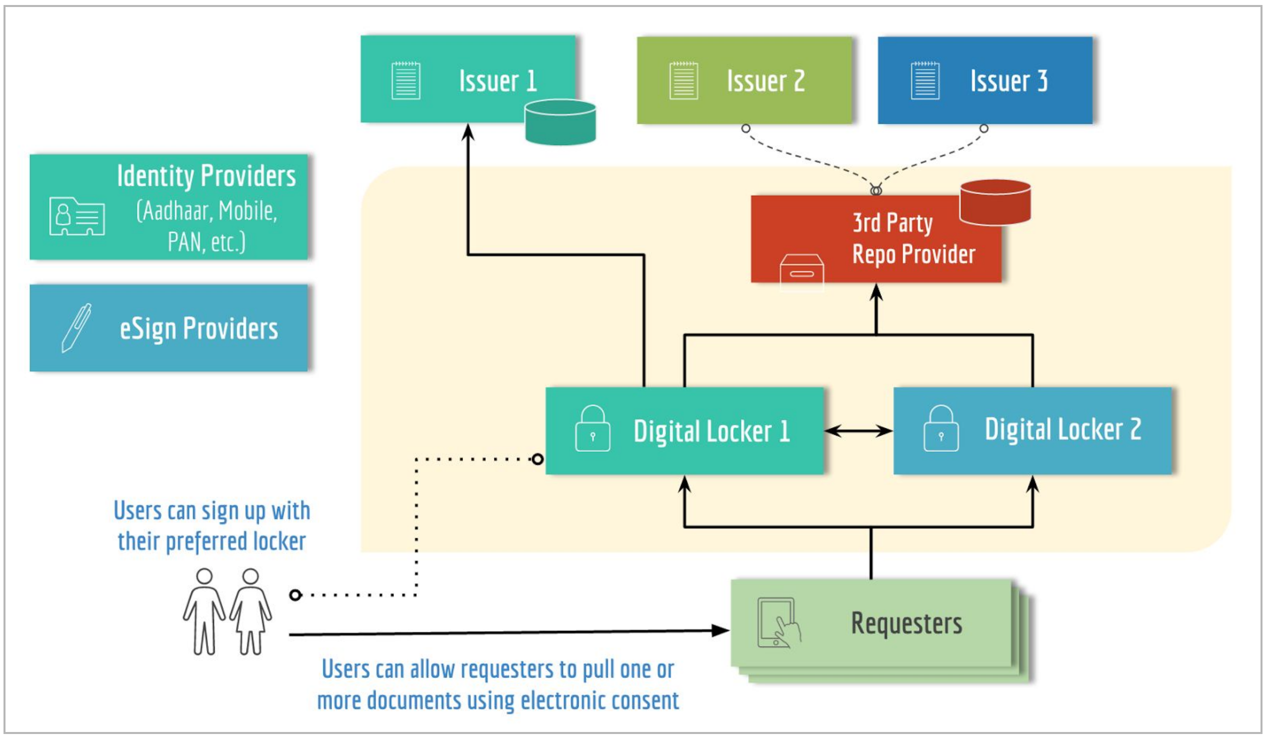
Assumptions

- Considering that various entities in both the public and private domain issue documents, solution should provide a scheme for multiple repositories to coexist and interoperate seamlessly. This also avoids a design requiring a single central database for all across the country.
- User documents in electronic formats are stored in federated fashion (no centralized single document repository). Designated set of central repositories would be part of the system (e.g. central repositories dedicated to universities and educational institutions, a dedicated repository for health domain, etc). Each entity or agency is responsible for issuing documents/certificates in electronic form and storing them in a designated repository.
- Departments/agencies/entities should be able to digitize older existing documents and bring them into this common electronic document system and provide seamless access.
- All documents in electronic formats must be digitally signed and timestamped by the issuing department/agency/entity to be trustable by other agencies and be compliant with IT Act.
- In future, it is highly recommended that all documents are issued against Aadhaar number(s) or similar strongly verifiable identity to ensure ONLY that person can indeed claim ownership. If a strong identity is not attached, a government document may be misused by people with same demographics (name/age/gender).
- All future electronic documents are available in both machine readable and printable formats. Old documents that are digitized may or may not have a corresponding machine readable form. This allows departments to easily start digitizing documents and gradually adopt a fully electronic form.

Proposed Architecture

This section covers solution architecture in detail including terminology used, high level architecture diagram, document identification scheme, document issuance lifecycle, document sharing scheme, and some examples.

In the diagram below, top side represents the issuance part and bottom side represents the real-time access part. Diagram depicts the federated model of document storage via designated dedicated digital repositories. Individuals can sign up for their preferred Digital Locker and use that to obtain consolidated view and also share documents with requesters using electronic consent.


Issuer 1
Issuer 2
Issuer 3
Identity Providers (Aadhaar, Mobile, PAN, etc.)
eSign Providers
3rd Party Repo Provider
Digital Locker 1
Digital Locker 2
Users can sign up with their preferred locker
Requesters
Users can allow requesters to pull one or more documents using electronic consent

Core Features

- Multiple electronic document repositories (either issuer's own or 2rd party under the contract with issuers) complying to DLTS specifications allowing rapid digitalization at the issuer level.
- Storage and access using unique document URIs and by Aadhaar numbers or similar strongly verifiable identity (access using a document URI points to one e-document while access via a verifiable identity(ies) can point to multiple e-documents issued to him/her).

    - All access via auditable and non-repudiable mechanisms.
    - All access within the ecosystem is with user consent. Note that issuers may directly expose documents that are public in nature (e.g., land registration or voter card) independent of the digital locker scheme.
    - Access via an Aadhaar number or similar universal identity must always be with user authentication to ensure access is authorized by that user (Identity holder).
- Various Issuers can issue new e-documents completely independently of other issuers at their own pace.
- Issuers can also choose to digitize older documents without having a machine readable representation and allow a verifiable and secure access to older (legacy) documents. They can focus on new e-documents in machine readable format while progressively providing digitization of older (existing) documents.
- Users can also choose to convert their existing documents using self signed mechanism and store them in their preferred Digital Locker. Acceptance of such documents will depend on the requester rules.
- Issuers may provide a printed copy of the document to the individual after storing them in their repository making the e-document verifiable, shareable, accessible, and re-printable.
- Issuers can choose their own print formats and styles as they do today.
- Issuers can use their own document numbering scheme and introduce new scheme for new documents without having to stick with one numbering scheme for ever.
- Issuer can choose one of the designated repository provider for all their documents or choose separate repository providers for different "document types".
- Users can share the document with other agencies/departments (e.g., sharing CBSE mark sheet with a university) just by sharing the document URI printed on their certificate instead of providing a photocopy or scanned copy.
    - Sharing via simple URI allows document sharing via mobile, SMS, website, etc easily.
    - In case where the document being shared is classified as PRIVATE or SECURE an explicit consent of the user is required prior to sharing with the requester.
- Requesters can dynamically obtain the list of document types defined by issuers by querying the metadata of the repository allowing issuers to add new document types dynamically.

- Requesters to use Digital Locker providers to access URIs of the required document. The requesters would get temporary access to the documents directly from the repositories for download.
- DLTS framework uses open standards and multi-provider ecosystem strategy. Both government and private players could play the role of repository and Digital Locker providers so that issuers and requesters get the best choice that fits their needs.
- DLTS framework addresses all core objectives as described earlier in this document while keeping entire system open via common interoperable standards.

It is important to note that interface between Repository and Digital Locker MUST BE via DLTS compliant APIs to ensure these are loosely coupled. This loose coupling allows providers to refactor, scale, administer, move, etc. independent of each other. If a provider offers both repository and Digital Locker features, it must be ensured that users (Issuers and Requesters) can sign up for individual services instead of bundling everything into a monolithic offering.

Unique Document URI

Every document that is issued and made accessible via DLTS system must have a unique way to resolve to the correct repository without conflict. This is critical to eliminate the need for all documents reference to be in one system. Federated repositories storing documents issued by various departments/agencies/entities must be reachable in a unique fashion.

All documents issued in compliance to DLTS should have the following URI format:

`IssuerID::DocType::DocID` where
- **`IssuerID`** is a unique issuer entity ID across the country
- **`DocType`** is the document type optionally defined by the issuer
- **`DocID`** is a unique document ID within the issuer system

Document URI MAY also have multiple alias defined to help one time use or auto-expire after a certain time limit as defined by the repositories. Alias feature can be used to hide the permanent Document URIs.

Issuer ID

`IssuerID` is (mandatory). All entities issuing e-documents, termed as Issuers must have a unique identification to ensure all documents issued by them are accessible via Digital Locker system.

It is recommended that list of unique issuer codes be derived via their domain URL whenever available and be published by the Digital Locker Authority with ability to add new issuers on need basis. When URL is not available for a department, a unique (alpha) code may be assigned.

Examples of issuer Ids are `maharashtra.gov.in` for Maharashtra State Government, `kseeb.kar.nic.in` for Karnataka School Board, `CBSE` for CBSE School Board, `UDEL` for Delhi University, etc. These codes MUST BE unique across India and managed by Digital Locker Authority.

Document Type

Issuers can freely define `documentType` values as per their internal classification. For example, CBSE may classify certificates into MSTN (10th mark sheet, KVPY (certificate issued to KVPY scholarship fellows), etc. Issuers should expose the details of their document types via meta APIs defined under this specification. Classifying documents into various types also allows issuers to choose different repositories for different types. This is to future proof the design without making assumption that all certificates issued by the issuer are available in same repository. This also allows migration from one repository to another in a gradual way. Issuers are free to define their document types without worrying any collaboration across other issuers.

Keeping the length minimal allows manual entry of document URI without making it too long. Hence it is recommended to keep length to be only up to 5. It is recommended that issuers define document types using pure alpha case-insensitive strings of length up to 5. These document types MUST BE unique WITHIN the issuer system.

Document ID

A `documentID` determined by the issuer should be assigned to every document. It MUST BE unique either within the document types of that issuer or it can be unique across all document types of that issuer.

Document ID is an alphanumeric string with recommended maximum length of 10. Document IDs MUST BE unique WITHIN the issuer system within a specific Document Type. Using random alphanumeric string eliminates the possibility of guessing next sequence number and accessing a list of documents in a sequential way without authorization.

It is highly recommended that issuer needing to issue a total of ***n*** documents within a document type use at least ***10n*** random space from which the strings/numbers are chosen to randomly allocate. Since issuers can easily add a new document type without any collaboration and approvals across other issuers, if more numbers are required, a new document type may be introduced.

Examples

Following are few examples of document URI printed on the document using QR code and a human readable string.

CBSE::MSTN::22636726	**KARMR::MCA::7385491**	**DLSSB::HSMS::GJSGEJXS**

Document Issuance Flow

Document issuance flow is given below:

1. Create a new digitally signed e-document with a unique URI. If the e-document is machine readable, provide an optional printable version.
2. Issuer should have tied up with a repository provider for storing documents and making it available online.
3. Store the new e-document in the repository of issuer choice.
4. Push the link to the e-document to issuer's Digital Locker provider against the user identity (such as Aadhaar).
5. Issuer may send email to user with a digitally signed e-document or provide a printed copy to the individual(s) for whom the document is issued to.
6. Document should have its URI, in the format **`issuerID::docType::docID`**, in both text and QR code form.

Older documents being digitized or documents that are issued purely in printable form (such as digitally signed PDFs) should be defined under published list of document types so that just by using document ID, corresponding URI can be dynamically formed and document accessed from its repository. This is achieved via a set of meta APIs defined in this specification.

Document Sharing and Access Flow

Users wanting to share their documents to requester agencies will have the following flow:

1. Requester (department/agency wanting access e-documents of the user) asks the user to provide the e-document URI. There are three ways for users to share e-documents with requesters:
    a. Share the document URI (since this is simply a text string, this can be collected easily). Requesters can pull the document via Digital Locker.
        i. For older documents with no DLTS compliant URI, requester can ask the user to choose the issuer and document types from a list, using the document type meta API, collect necessary document identifiers, and then internally form URI.
    b. Or use the Digital Locker provider of the requester through which user locker is accessed using user authentication, obtain document URI list (including self-signed documents stored in user locker), select the URIs required by the requester, and obtain user consent to pull them from various destinations via DLTS APIs.
2. Requester uses their Digital Locker provider to access e-documents stored across federated DLTS repositories.
3. Requester uses DLTS FetchDoc API to access the e-documents based on the URI.
    a. Note that some document types of some of the issuers may require online user authentication (via Aadhaar or Aadhaar enabled authentication provider such as e-Pramaan). This is based on Issuer rules.
    b. If the document type requires document owner authentication, appropriate authentication credentials (biometric, OTP, etc.) also may have to be captured.
4. Digital Locker system looks up its internal database and maps the URI to an actual repository URL and forwards the request to the appropriate repository.

    a. Digital Locker provider must validate credentials of Requester (API license key, digital signature, etc) and only on successful validation, request can be forwarded to repository.
    b. Digital Locker sends the request to repository based on the DLTS Repository API.
5. Repository provider returns the digitally signed document after authentication of requester by Digital Locker system.
    a. Repository provider must validate credentials of Digital Locker (API license key, digital signature, etc)
    b. If the document type requires authenticated access, before returning the document, it must verify the authentication response provided by the Digital Locker.
6. E-Document returned by the repository goes back to requester via Digital Locker. For large printable documents, a temporary printable URL can be provided so that requester can directly download from repository without going via Digital Locker (this must be one time download URL that expires once download is done).
7. Digital Locker must not store authentication credentials as-is. Necessary audit must be stored. If consent requires notification of document access to a destination, requester's Digital Locker must ensure all notifications are sent.
8. Requester uses the e-Document for its purposes and provides the intended service to the user.

Security & Privacy Aspects

1. Document Security
    a. Proposed solution eliminates the need for all e-documents to be stored in one central repository to minimize the risk of security and availability.
    b. Since e-documents are mandatorily digitally signed and timestamped, NO alteration can be done for misuse.
2. Access Security
    a. Repository and Digital Locker providers must comply with DLTS security and API specifications.
    b. While some e-document types may be available to trusted requesters without electronic authentication and authorization of the owner, some document types may mandatorily require explicit electronic authentication and authorization of the document owner for every access.
    c. Access to repositories is only protected by API license keys, digital signature, secure transport, and access level audits.

d. If explicit authentication is required for a particular document type, a trusted authentication of the owner is mandated during access.
    i. Architecture allows owner authentication via Aadhaar or other trusted 3rd party authentication schemes.

e. Mandatory audit logs must be maintained both by Digital Locker and repository providers.

f. Repository provider MUST publish anonymized access logs in public for transparency as well as notification subscriptions by the document owners.

E-Document Specifications

Document URI

All documents issued in compliance to DLTS should have the following URI format:

`<IssuerID>::<DocType>::<DocID>`

Where,

- **`IssuerID`** (mandatory) - is a unique issuer entity ID. This is a unique pure alpha case-insensitive string. Digital Locker Authority shall assign and maintain a list of unique issuer IDs to ensure they are globally unique. To easily make it unique, department's domain URL can be used whenever available or the Digital Locker Authority may assign a code.
- **`DocType`** (optional) - is the document type optionally defined by the issuer. This is highly recommended for document classification and versioning purposes. Issuers may decide their own classification mechanism.
- **`DocID`** (mandatory) - is a alphanumeric document ID that is unique within a document type of the issuer system. It is highly recommended that issuers use random strings to avoid guessing the sequence numbers of document IDs.

Document Owner

For avoiding document misuse, it is critical that all e-documents are attached to one or more strongly verifiable identities such as Aadhaar. For example, a caste certificate may be attached to one Aadhaar holder while a marriage certificate is attached to two Aadhaar holders. DLTS solution offers a mechanism for issuers to secure access via Aadhaar authentication of any of the owners.

Document Format

It is highly recommended that issuers use machine readable formats such as XML/JSON for new e-documents issued by them and also provide a printable version. This ensures requester system process flows to be fully automated. It is also important to note that DLTS solution does not insist on a common printing format allowing issuers to freely choose a print format for their document. Use of logo, colors, etc are freely possible within the print format.

All e-documents **MUST BE digitally signed** by the issuer to ensure they are verifiable for authenticity and are unalterable. This also ensures all fake documents are eliminated across the system.

Documents that have strict time bound information should use timestamped digital signatures. This is optional for other documents. The issuer must ensure the safety of digital signature a compromise would result in publishing invalid certificates and as IT Act the owner of the certificate is responsible for such events. If issuers are re-issuing the documents for content alteration or revocations can use **pushUri** API to notify the locker.

API Specifications

A detailed set of API and compliance specifications have been prepared. These API's cover the all functions of a service providers within the Digital Locker ecosystem. These include:

1. **PushURI**: Allow Issuers having strongly verifiable identity (e.g. Aadhaar) seeded documents to push URI's of these documents directly into the Digital Locker of the issuer or account of the User.
2. **PullDoc**: Allows User to pull a document from the Issuer repository into the Digital Locker by providing a URI or pull a URI of a document by searching for his/her own document from the repository of the Issuer.
3. **FindURI**: Allows the user to get all the URIs from various Digital Lockers that are attached to user's universal Identities such as Aadhaar.
4. **FetchDoc**: Allows a Requestor to fetch a document for a given URI after having received the user consent.
5. **GetLockerToken:** and the schema definition of the **Consent Token** – Electronic consent artefact created and audited as per MeitY's Electronic Consent framework.

The following Meta APIs facilitate seamless integration between ecosystem partner applications:

1. **getIssuers**: Allows ecosystem partners to fetch all the Issuers registered with the regulator to provide Digital Locker services.
2. **getLockers:** Allows ecosystem partners to fetch all the Locker Providers registered with the regulator.
3. **getDocumentTypes**: Allows ecosystem partners to fetch all the applicable Document Types supported by the systems. For e.g, document types are represented by unique alpha based DocumentTypeId of maximum length 5 along with a short desc of the document - "Birth Certificate", "Marriage Certificate", "Employment Certificate" etc.,
4. **getDocumentLookupAttributes**: Each Issuer shall map the documents with unique user attributes for retrieval. For e.g CBSE maps the documents using Name, Roll Number and Year. This API enables Digital Locker providers to easily identify the lookup parameters and to pull the documents from Issuers repository. With the saturation of Aadhaar seeding across issuers repository, Adhaar may eventually become one such uniquely identifiable parameter.

IMPORTANT: **All API requests and responses must be digitally signed** by the respective organization initiating the APIs. Digital Signature XML element must adhere to the W3C standards - https://www.w3.org/TR/xmldsig-core/
https://tools.ietf.org/html/rfc7515

PushURI API Specification

The REST based PushURI API can be used to push a single URI into Digital Locker. This API can be used to push the document details to Digital Locker as and when a document is generated in the issuer system.

Request URL (API End Point)

```
https://<hostname+port>/LOCKER/:code/:ver/PushUri
```

METHOD POST

HEADER
Name: Content-Type

Value: application/xml

XML REQUEST STRUCTURE

```
<?xml version="1.0" encoding="utf-8"?>
<PushUriRequest ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" issuerId=""
lockerId="" keySign="AES256_GCM(api_key,ts+random-no)" keyIndex="" >
    <Uri alias="" value="">
        <IssuedTo>
            <Id type="AADHAAR|MOBILE|PAN|..." val="" />
        </IssuedTo>
        <IssuedOn>DD/MM/YYYY</IssuedOn>
        <ValidFrom>DD/MM/YYYY</ValidFrom>
        <ValidTo>DD/MM/YYYY</ValidTo>
     </Uri>
     <Signature> Digital Signature of the Issuer </Signature>
</PushUriRequest>
```

Please refer to Appendix-1 for description of common request attributes. Various elements/attributes in the request are described below.

Sr. No.	XML Element	Mandatory / Optional	Description
1.	`issuerId`	M	Issuer Id initiating the PushUri API request.
2.	`lockerId`	M	Locker Id to which the Uri is being pushed to.
3.	`Uri`	M	This element contains the details of the Document Uri being push to the locker. Please refer to Appendix section for additional element details.
4.	`Signature`	M	Digital signature of the Issuer signing the entire request XML. The CN value of the Issuer's Digital Signature MUST match with the domain name of the Issuer.

XML RESPONSE STRUCTURE

The response to the Push URI Request will include return value of Y or N. If failed to process by the digital locker system detailed error code is shared. The XML response structure is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<PushUriResponse ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" respCode=""
issuerId="" lockerId="" ret="Y|N" err="" errMsg="">
       <Signature> Signature of the Locker Provider </Signature>
</PushUriResponse>
```

Please refer to Appendix-1 for description of common response attributes. Various elements/attributes in the response are described below.

Sr. No.	**XML Element**	**Mandatory (M)/ Optional (O)**	**Description**
1.	`issuerId`	M	Issuer Id initiating the PushUri API request.
2.	`lockerId`	M	Locker Id to which the Uri is being pushed to.
3.	`Signature`	M	Digital signature of the locker signing the entire response XML.

PullDoc API Specification

The REST based Pull Doc Request API has to be implemented by the issuers and will be consumed by Digital Locker system. This API may be invoked when the user clicks on the URIs displayed within the Digital locker portal/application. The issuer system will respond to this API by sending the document data. The document data can be sent in any open format depending on the request send by Digital Locker. Common formats like XML, JSON, PDF, DOCX, XLSX, JPG, PNG, etc. may be used.

Request URL (API End Point)
`https://<hostname+port>/ISSUER/:code/:ver/PullDoc/`

METHOD GET

HEADER
Name: Content-Type
Value: application/xml

XML REQUEST STRUCTURE

The following is the XML request template for the PullDoc Request API.

```
<?xml version="1.0" encoding="UTF-8"?>
<PullDocRequest ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" issuerId=""
lockerId="" keySign="AES256_GCM(api_key,ts+random-no)" keyIndex="">
       <DocOwner>
              <Id type="AADHAAR|MOBILE|PAN|..." val="" />
              <Details name="" dob="DD/MM/YYYY" mobileNum="" />
       </DocOwner>
       <DocDetails consentId="">
              <DocUri></DocUri>
              <UDFS docType="" docContent="Y|N">
                     <!-- following element is repeating -->
                     <UDF key=""> UDF Value </UDF>
              </UDFS>
       </DocDetails>
       <Signature> Digital Signature of locker provider </Signature>
</PullDocRequest>
```

Please refer to Appendix-1 for description of common request attributes. Various elements/attributes in the request are described below-

Sr. No.	XML Element {Attribute}	Mandatory/ Optional	Description
1.	`lockerId`	M	Locker Id initiating the PullDoc API request.
2.	`issuerId`	M	Issuer Id to which the Document is being pushed to.
3.	`DocOwner`	M	Please refer to Appendix for details of DocOwner element. This element is for audit purposes to know who the requester is for the issuer systems.
4.	`DocDetails`	M	One of the Uri or UDF element is Mandatory to lookup the document based on the uniquely identifiable Uri or lookup attributes as defined by the issuer system for a given documentType.

5.	`DocDetails {consentId}`	O	Consent ID of the electronic consent artefact provided by the user to Digital Locker to access the document.
6.	`DocDetails:URI`	O	URI identifies the document uniquely. **<u>NOTE</u> ONLY One value between URI or UDFS has be to provided in the request.**
7.	`DocDetails:UDFS {docType}`	O	If Uri is NOT present, docType refers to the Document Type Id being pulled. Each Document Type on the issuer MAY have different User Defined Fields to lookup the document.
8.	`DocDetails:UDFS {docContent}`	O	Possible values of this attribute are “Y” or “N”. If the value of this attribute is “N”, then the API must return the document metadata in XML format in the response. If the value of this attribute is “Y” (or attribute is not present in the request), then the API must return Base64 encoded signed PDF data in the response.
9.	`UDF {key}`	O	UDF Key is a User Defined Key provided by the Issuer. In case of CBSE the UDF Keys might be Roll Num, Class or Year. **<u>NOTE</u> ONLY One value between URI or UDFS has be to provided in the request.**
10.	`UDF`	O	The Value provided by user for a particular UDF Key. In case of CBSE Roll Number, it will be roll number of the user. ***NOTE***: Issuer SHOULD verify the User Defined Values against the DocOwner attributes passed in the request.
11.	`Signature`	M	Digital signature of the locker provider signing the entire request XML. The CN value of the locker provider’s Digital Signature MUST match with the domain name of the requesting locker.

XML RESPONSE STRUCTURE

The following is the XML response template for the Pull URI Response API.

```
<?xml version="1.0" encoding="utf-8"?>
<PullDocResponse   ver="1.0"   ts="YYYY-MM-DDThh:mm:ssZ+/-n.n"   txn=""   respCode=""
issuerId="" lockerId="" ret="Y|N" err="" errMsg="">
  <Document consentId="">
       <Uri alias="" value="">
              <IssuedTo>
                     <Id type="AADHAAR|MOBILE|PAN|..." val="" />
              </IssuedTo>
              <IssuedOn>DD/MM/YYYY</IssuedOn>
               <ValidFrom>DD/MM/YYYY</ValidFrom>
               <ValidTo>DD/MM/YYYY</ValidTo>
       </Uri>
       <DocContent Type="any valid mimetype" XSLTPath="">
           Base64 encoded of the document in various formats
       </DocContent>
  </Document>
  <Signature> Digital Signature of the Issuer </Signature>
</PullDocResponse>
```

Please refer to Appendix-1 for description of common response attributes. Various elements/attributes in the response are described below-

Sr. No.	XML Element	Mandatory/ Optional	Description
1.	`lockerId`	M	Locker Id initiating the PullDoc API request.
2.	`issuerId`	M	Issuer Id to which the Document is being pushed to.
3.	`Document`	M	Please refer to Appendix-1 section for more details about the Document element and its attributes.
4.	`Signature`	M	Digital signature of the locker provider signing the entire response XML.

FindURI API Specification

The REST based FindURI Request API has to be implemented by all the Digital Locker portal providers and will be consumed by other Digital Locker portal providers and 3rd party requesting entities.

This API will be invoked when the user opens a Locker account with a Digital Locker portal provider and this provider invokes this API call to obtain all the URI's of that user that may be hosted by other Digital Locker portal providers. The user must have an account with the other digital locker providers and obtain a token to initiate this call.

Alternatively, a 3rd party requesting entity (e.g., passport office system), May initiate this call to fetch the URIs linked to the user with a given digital locker provider.

The responding Digital Locker portal provider will respond to this API by sending the list of all the user's URIs to the requesting entities.

Request URL (API End Point)

```
https://<hostname+port>/LOCKER/code/:ver/FindURI/
```

METHOD GET

HEADER
Name: Content-Type
Value: application/xml

XML REQUEST STRUCTURE

```
<?xml version="1.0" encoding="utf-8"?>
<FindURIRequest ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" requesterId=""
lockerId="" keySign="AES256_GCM(api_key,ts+random-no)" keyIndex="" >
  <DocOwner>
       <Id type="AADHAAR|MOBILE|PAN|..." val="" />
       <Details name="" dob="DD/MM/YYYY" mobileNum="" />
  </DocOwner>
  <LockerDetail userId="" token="" />
  <Consent id="" />
  <Signature> Digital signature of the requesting entity </Signature>
</FindURIRequest>
```

Please refer to Appendix-1 for description of common request attributes. Various elements/attributes in the request are described below-

Sr. No.	XML Element	Mandatory/ Optional	Description
1.	`requesterId`	M	Requester Id initiating the findUri request API call. Requests can be other digital locker providers or 3rd party requesting entities like Passport Office, Bank etc.
2.	`lockerId`	M	Locker Id to which the findUri call is made to get the URIs associated to the user through 3rd party requesting entity.
3.	`DocOwner`	M	Please refer to Appendix for details of DocOwner element. This element is for audit purposes to know who the requester is for the issuer systems.
4.	`LockerDetail`	M	LockerDetail element captures the token value obtained in GetLockerTokenRequest API call to allow requesting entity to access user digital locker.
5.	`LockerDetail {userId}`	M	userId in the format of my-id@locker.gov.in in Virtual Personal Address (VPA) format.
6.	`LockerDetail {token}`	M	Token obtained by getLockerTokenRequest API call. ***Note*** - No user credentials to be collected by the requesting applications.
7.	`Signature`	M	Digital signature of the requesting entity of the complete request xml. The CN value of the requesting entity's Digital Signature MUST match with the domain name of the requester.

XML RESPONSE STRUCTURE

The following is the XML response template for the Find URI Response API.

```
<?xml version="1.0" encoding="utf-8" ?>
<FindURIResponse ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" respCode=""
requesterId="" lockerId="" ret="Y|N" err="" errMsg="">
    <URIs>
             <Uri alias="" value="">
             <IssuedTo>
                    <Id type="AADHAAR|MOBILE|PAN|..." val="" />
             </IssuedTo>
             <IssuedOn>DD/MM/YYYY</IssuedOn>
             <ValidFrom>DD/MM/YYYY</ValidFrom>
             <ValidTo>DD/MM/YYYY</ValidTo>
             </Uri>
    <URIs>
    <Signature> Digital signature of the digital locker provider </Signature>
</FindURIResponse>
```

Please refer to Appendix-1 for description of common response attributes. Various elements/attributes in the response are described below-

Sr. No.	XML Element	Mandatory/ Optional	Description
1.	`requesterId`	M	Requester Id initiating the findUri request API call. Requests can be other digital locker providers or 3rd party requesting entities like Passport Office, Bank etc.,
2.	`lockerId`	M	Locker Id to which the findUri call is made to get the URIs associated to the user through 3rd party requesting entity.
3.	`URIs`	M	List of URIs matched against the digital locker user.
4.	`Uri`	M	Please refer to Appendix-1 for more details about Uri element.
5.	`Signature`	M	Digital signature of the locker signing the entire response xml.

FetchDoc API Specification

Request URL (API End Point)

```
https://<hostname+port>/LOCKER/code/:ver/FetchDoc/
```

METHOD GET

HEADER
Name: Content-Type
Value: application/xml

XML REQUEST STRUCTURE

```
<?xml version="1.0" encoding="utf-8"?>
<FetchDocsRequest ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" requesterId=""
lockerId="" keySign="AES256_GCM(api_key,ts+random-no)" keyIndex=""  >
  <DocOwner>
       <Id type="AADHAAR|MOBILE|PAN|..." val="" />
       <Details name="" dob="DD/MM/YYYY" mobileNum="" />
  </DocOwner>
  <LockerDetail userId="" token="" />
  <Consent id=""> base64 encoded signed electronic consent artefact </Consent>
  <URIs>
       <Uri alias="" value="" />
  </URIs>
  <Signature>Digital signature of the requesting entity </Signature>
</FetchDocsRequest>
```

Please refer to Appendix-1 for description of common request attributes. Various elements/attributes in the request are described below-

Sr. No.	XML Element	Mandatory/ Optional	Description
1.	`requesterId`	M	Requester Id initiating the findUri request API call. Requests can be other digital locker providers or 3rd party requesting entities like Passport Office, Bank etc.,
2.	`lockerId`	M	Locker Id to which the findUri call is made to get the URIs associated to the user through 3rd party requesting entity.

3.	`DocOwner`	M	Please refer to Appendix for details of DocOwner element. This element is for audit purposes to know who the requester is for the issuer systems.
4.	`LockerDetail`	M	LockerDetail element captures the token value obtained in GetLockerTokenRequest API call to allow requesting entity to access user's digital locker.
5.	`LockerDetail {userId}`	M	userId in the format of my-id@locker.gov.in in Virtual Personal Address (VPA) format.
6.	`LockerDetail {token}`	M	Token obtained by getLockerTokenRequest API call. ***Note*** - No user credentials to be collected by the requesting applications.
7.	`Consent`	M	Consent element represent the signed consent artefact of the digital locker user requesting to access his/her document through the 3rd party requesting entity. This element hold base64 encoded signed electronic consent of the digi locker user.
8.	`Consent {id}`	M	Electronic consent artefact id.
9.	`URIs`	M	List of URIs for which the documents need to be fetched for the requesting digital locker user using 3rd party requester entity.
10.	`Uri`	M	Please refer to Appendix-1 for more details about Uri element. NOTE: This request will require only the Uri alias and/or Uri value for fetching the documents. ***The other elements for the Uri element if present in the request are ignored.***
11.	`Signature`	M	Digital signature of the requesting entity of the complete request xml. The CN value of the requesting entity's Digital Signature MUST match with the domain name of the requester.

XML RESPONSE STRUCTURE

The XML response structure for FetchDocs request is as follows:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<FetchDocsResponse ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" respCode=""
requestId="" lockerId=""  ret="Y|N" err="" errMsg="" >
   <Documents>
         <Document>
             <Uri alias="" value="">
               <IssuedTo>
                    <Id type="AADHAAR|MOBILE|PAN|..." val=" />
               </IssuedTo>
               <IssuedOn>DD/MM/YYYY</IssuedOn>
               <ValidFrom>DD/MM/YYYY</ValidFrom>
               <ValidTo>DD/MM/YYYY</ValidTo>
                </UriDetails>
                <DocContent Type="PDF|XML" xsltPath="">
                     Base64 encode of signed PDF/XML document
                </DocContent>
          </Document>
   </Documents>
   <Signature />
</FetchDocsResponse>
```

Please refer to Appendix-1 for description of common response attributes. Various elements/attributes in the response are described below-

Sr. No.	XML Element	Mandatory/ Optional	Description
1.	`requesterId`	M	Requester Id initiating the findUri request API call. Requests can be other digital locker providers or 3rd party requesting entities like Passport Office, Bank etc.,
2.	`lockerId`	M	Locker Id to which the findUri call is made to get the URIs associated to the user through 3rd party requesting entity.
3.	`Documents`	M	Please refer to Appendix-1 for more details about the Documents element and its attributes.
4.	`Signature`	M	Digital signature of the entire xml response signed by the locker provider.

GetLockerToken API Specification

The REST based GetLockerToken API has to be implemented by all the Digital Locker portal providers. This API will be consumed by the other Digital Locker portal providers or registered 3rd party requesting entities (for e.g., passport office, banks etc.,) to give a seamless experience to any digital locker users to navigate and pull the documents.

Digital Locker user should never share his/her digital locker credentials to the requesting entities. This API will be invoked by the registered 3rs party entities to get the authorization token to access user's digital locker using one of the following ways:

1. User shares his/her virtual user-id of digital locker account along with the One Time Password issued by the digital locker provider's app.
   a. Verification of the One Time Password shall generate the authorization token.
2. User shares his virtual user-id of the digital locker along with a collect flag set to "Y".
   a. Digital Locker provider notifies the user through the digital locker app and collects authorization using simple yes or no interface.
   b. Digital Locker provider should provide additional information about the requesting entity and Document URIs being accessed.

Request URL (API End Point)

```
https://<hostname+port>/LOCKER/:code/:ver/GetLockerToken/
```

METHOD GET

HEADER
Name: Content-Type
Value: application/xml

XML REQUEST STRUCTURE

```
<GetLockerTokenRequest ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn=""
requesterId="" lockerId="" keySign="AES256_GCM(api_key,ts+random-no)" keyIndex="">
       <LockerAccess userId="" collectToken="Y|N" otp="" />
       <URIs>
             <!-- following element is repeating -->
```

```
            <uri alias="" value="" />
        </URIs>
        <Signature> Digital signature of the 3rd party requesting entity </Signature>
</GetLockerTokenRequest>
```

Please refer to Appendix-1 for description of common request attributes. Various elements/attributes in the request are described below-

Sr. No.	XML Element	Mandatory / Optional	Description
1.	`requesterId`	M	Requester Id initiating the GetLockerToken request API call. Requests can be through other digital locker providers or 3rd party requesting entities like Passport Office, Bank etc.,.
2.	`lockerId`	M	Locker Id to which the findUri call is made to get the URIs associated to the user through 3rd party requesting entity.
4.	`LockerAccess`	M	LockerAccess element captures the locker userId info in VPA format along with other access options.
5.	`LockerAccess {userId}`	M	userId in the format of my-id@locker.gov.in in Virtual Personal Address (VPA) format.
6.	`LockerAccess {collectToken }`	M	If collectToken attribute is 'Y' Digital Locker shall alert the user through mobile app or other channel to get the user consent to grant access to the documents.
7.	`LockerAccess {otp}`	M	otp attribute hold the One Time Password to access the digital locker to fetch URIs or Documents from the 3rd party requesting entities.
9.	`URIs`	M	List of URIs for which the documents need to be fetched using 3rd party requesting entity. The list of URIs in the request will enable the Digital Locker to alert the user

			while collecting the Token or granting access through OTP.
10.	`Uri`	M	Please refer to Appendix-1 for more details about Uri element. NOTE: This request will require only the Uri alias and/or Uri value for fetching the documents. ***The other elements for the Uri element if present in the request are ignored.***
11.	`Signature`	M	Digital signature of the requesting entity of the complete request xml. The CN value of the requesting entity's Digital Signature MUST match with the domain name of the requester.

XML RESPONSE STRUCTURE

The XML response structure for GetLockerToken response is as follows:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<GetLockerTokenResponse ver="1.0" ts="YYYY-MM-DDThh:mm:ssZ+/-n.n" txn="" respCode=""
orgId="" ret="Y|N" err="" errMsg="" >
       <LockerToken userId=""  token="" expiry="YYYY-MM-DDThh:mm:ssZ+/-n.n" />
       <Signature />
</GetLockerTokenResponse>
```

Please refer to Appendix-1 for description of common response attributes.

Sr. No.	XML Element	Mandatory / Optional	Description
1.	`requesterId`	M	Requester Id initiating the GetLockerToken request API call.
2.	`lockerId`	M	Locker Id to which the GetLockerToken API call is made to get the LockerToken to access URIs in the request.
3.	`LockerToken`	M	Locker Token to grant access to URI documents.

			IMPORTANT: The Digital Locker system MUST hold a reference to the URIs in the request, requestorId, against the Token values and ONLY release the documents in the FetchDocs API call. Only One token must be active for a given digital locker userId & requesterId combination.
4.	`LockerToken {userId}`	M	userId in the format of my-id@locker.gov.in in Virtual Personal Address (VPA) format.
	`LockerToken {token}`	M	Unique token value assigned to the user for against the
	`LockerToken {tokenExpiry}`	M	Token expiry time (`YYYY-MM-DDThh:mm:ssZ+/-n.n`) set by the digital locker system. This time can be derived as minimum of digital locker user preferences or digital locker's default value.
4.	`Signature`	M	Digital signature of the entire xml response signed by the locker provider.

Meta APIs Specification

Get Issuers

`https://<hostname+port>/LOCKER/:code/:ver/issuers/`

XML RESPONSE STRUCTURE

```
<Issuers publishedTs="" >
      <!-- following element is repeating -->
       <Issuer id="" url="" name="">
             <LocalNames>
                 <!-- following element repeats for all official languages -->
                 <LocalName langCode="" value=""/>
             </LocalNames>
             <KeyInfo>
                    <X509Data> <!-- two pointers to certificate-A -->
                         <X509IssuerSerial>
                           <X509IssuerName>CN=kseeb.kar.nic.in, OU=kseeb,
                  O=kseeb,
                             L=Bangalore, ST=Karnataka, C=IN</X509IssuerName>
                           <X509SerialNumber>99999979</X509SerialNumber>
```

```
                          </X509IssuerSerial>
                          <X509SKI>45a75ca4</X509SKI>
                     </X509Data>
                     <X509Data><!-- single pointer to certificate-B -->
                          <X509SubjectName>Subject of Certificate B</X509SubjectName>
                     </X509Data>
                     <X509Data> <!-- certificate chain -->
                          <!--Signer cert, issuer CN=ApprovedCA,OU=CER,O=IndCA,C=IN,
                   serial 4-->
                          <X509Certificate>INDaCCA..</X509Certificate>
                          <!-- Intermediate cert subject
                   CN=ApprovedCA,OU=CER,O=IndCA,C=IN
                               issuer CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
                          <X509Certificate>INDCCA...</X509Certificate>
                          <!-- Root cert subject CN=CCA,OU=Issuer,O=CCA,C=IN -->
                          <X509Certificate>MIICSTCCA...</X509Certificate>
                     </X509Data>
              </keyInfo>
       </Issuer>
<Issuers>
```

Get Lockers

https://<hostname+port>/LOCKER/:code/:ver/lockers/

XML RESPONSE STRUCTURE

```
<Lockers publishedTs="" >
      <!-- following element repeats -->
       <Locker id="" url="" name="">
              <Issuers>
                    <!-- following element repeats -->
                     <Issuer id="" />
              </Issuers>
             <KeyInfo>
                     <X509Data> <!-- two pointers to certificate-A -->
                          <X509IssuerSerial>
                            <X509IssuerName>CN=kseeb.kar.nic.in, OU=kseeb,
                   O=kseeb,
                              L=Bangalore, ST=Karnataka, C=IN</X509IssuerName>
                            <X509SerialNumber>99999979</X509SerialNumber>
                          </X509IssuerSerial>
                          <X509SKI>45a75ca4</X509SKI>
                     </X509Data>
                     <X509Data><!-- single pointer to certificate-B -->
                          <X509SubjectName>Subject of Certificate B</X509SubjectName>
                     </X509Data>
                     <X509Data> <!-- certificate chain -->
                          <!--Signer cert, issuer CN=ApprovedCA,OU=CER,O=IndCA,C=IN,
                   serial 4-->
                          <X509Certificate>INDaCCA..</X509Certificate>
                          <!-- Intermediate cert subject
                   CN=ApprovedCA,OU=CER,O=IndCA,C=IN
                               issuer CN=tootiseCA,OU=FVT,O=Bridgepoint,C=US -->
                          <X509Certificate>INDCCA...</X509Certificate>
                          <!-- Root cert subject CN=CCA,OU=Issuer,O=CCA,C=IN -->
```

```
                         <X509Certificate>MIICSTCCA...</X509Certificate>
                    </X509Data>
             </keyInfo>
             <LocalNames>
                 <!-- following element repeats for all official languages -->
                 <LocalName langCode="" value=""/>
            </LocalNames>
<Lockers>
```

Get DocumentTypes

`https://<hostname+port>/LOCKER/:code/:ver/:issuerId/documentTypes/`

XML RESPONSE STRUCTURE

```
<DocumentTypes publishedTs="" >
       <Issuer id="" />
       <!-- default access level is PRIVATE -->
       <DocumentType id="" name="" accessLevel="CONFIDENTIAL|PRIVATE|PUBLIC">
             <LocalNames>
                 <!-- following element repeats for all official languages -->
                 <LocalName langCode="" value=""/>
             </LocalNames>
        </DocumentType>
</DocumentTypes>
```

Get DocumentTypeAttributes

`https://<hostname+port>/LOCKER/:code/:ver/documentTypeAttributes/:issuerId/:documentTypeId/`

:issuerId - issuerId for which documentType look up is requested.
:documentTypeId - doucmentTypeId for which user defined fields are required

XML RESPONSE STRUCTURE

```
<issuer id="" publishedTs="">
       <DocumentType  id="" >
             <UDFs>
                    <UDF> first look up of issuer’s user defined key </UDF>
                    <UDF> second look up issuer’s user defined key </UDF>
             <UDFs>
       </DocumentType>
<Issuer>
```

APPENDIX - I

Common XML Element Definitions

1. Common Request Element Attributes

1.	`ver`	M	API version. Must match with the version in the request URL. Currently supported version is 1.0
2.	`ts`	M	A timestamp value. This will be used to decode the keySign element described below. YYYY-MM-DDThh:mm:ssz+/-n.n (derived from ISO 8601). Time Zone should not be specified and is automatically defaulted to IST (UTC +5:30).
3.	`txn`	M	A ***unique*** transaction id provided by calling entity of the API request. This will be used to uniquely identify the request by the service endpoint. It is recommended the requesting entity application prefix a unique sequence.
5.	`keySign`	M	Provide encrypted value (using AES_GCM) of the xml/JSON elements with API key provided to the requesting entity with timestamp value and 12 digit random number concatenated together in this sequence. You must use the same timestamp value that you have specified in the ts element described above.
6.	`keyIndex`	M	Valid API key index used in keySign. Key index should be number. Requesting entities can have multiple API keys issued over time and index helps to easily identify the API key used for a given request.

2. Common Response Element Attributes

1.	`ts`	M	A timestamp value as sent in the response.
2.	`txn`	M	Transaction id as passed in the request.

3.	`respCode`	M	Unique response code for every transaction processed by the service endpoint.
4.	`ret`	M	Y in case of success and N in case of error.
5.	`err`	O	If ret is 'N', error code to describe the error.
6.	`errMsg`	O	Appropriate error message in case of an error.

3. DocOwner

Document Owner element captures document owner details requesting the documents with locker or issuers for audit purposes.

```
<DocOwner>
	<Id type="AADHAAR|MOBILE|PAN|..." val="" />
	<Details name="" dob="DD/MM/YYYY" mobileNum="" />
</DocOwner>
```

1.	`Id {type}`	M	Type of user's ID used while issuing the document by the Issuer. Eg. "AADHAAR" for Aadhaar number or "MOBILE" for mobile number or "PAN" for Pan number of the document owner.
2.	`Id {val}`	M	The Value of the ID used. For e.g., In case of AADHAAR as ID Type then Aadhaar Number of the user.
3.	`Details {name}`	M	Full Name of the user on the Digital Locker portal.
4.	`Details {dob}`	M	Date of Birth of the user.
5.	`Details {mobileNum}`	O	Mobile number of the user.

4. Document

Document element represent the complete meta information about the document and the document content.

```
<Document consentId="" langCode="">
	<Uri alias="" value="">
		<IssuedTo>
```

```
                    <!-- following element is repeating -->
                     <Id type="" val="" />
              </IssuedTo>
              <IssuedOn>DD/MM/YYYY</IssuedOn>
              <ValidFrom>DD/MM/YYYY</ValidFrom>
              <ValidTo>DD/MM/YYYY</ValidTo>
       </Uri>
       <DocContent Type="" xsltPath="">
           Base64 encode of signed document in various formats
       </DocContent>
</Document>
```

1.	`Document`	M	Represents Documents metadata information in UriDetails sub-element and actual content within DocContent sub-element. Document element repeats itself under Documents tag to share more than one document.
2.	`Document {consentId}`	M	Consent Id used to retrieve the document. Consent Id uniquely identifies electronic consent of the document owner.
3.	`Uri`	M	This section will contain the User to whom the document has been issued and the document details - meta info and doc content.
4.	`Uri {alias}`	M	Short URI of the documents as stored in issuer repository. This short URI will be used to fetch a document from the issuer repository. **Uri alias can be a one time sharable link for sensitive documents with an optional expiry time set by the Issuer/locker.**
5.	`Uri {value}`	M	Complete URI of the documents as stored in issuer repository. This Complete URI will be used to fetch a document from the issuer repository.
6.	`IssuedTo`	M	Contains the details of the User to whom this document has been issued by the issuer. There can be multiple owners of the same document. For e.g Marriage Certificates, Sale Deeds etc.
7.	`Id {type}`	M	Type of user’s ID used while issuing the document by the Issuer. Eg. “AADHAAR” for

			Aadhaar number or “MOBILE” for mobile number of the user, etc
8.	`Id {val}`	M	The Value of the ID used. For e.g In case of type is “AADHAAR” then Aadhaar Number of the user.
9.	`IssuedOn`	M	The issue date of the document in DD/MM/YYYY format.
10.	`ValidFrom`	M	The date from which the document is valid in DD/MM/YYYY format. This may be same as the issue date. In case not applicable please put a blank value.
11.	`ValidTo`	M	The expiry date of the document in DD/MM/YYYY format. In case not applicable please put a blank value.
12.	`Document: DocContent {type}`	M	The format of the document sent in the request. Value may be PDF or XML or other valid formats. Standard mime type should be used.
13.	`Document: DocContent {xsltPath}`	O	**Mandatory** if type is XML. XSLT Path to help render the XML document in a printable format.
14.	`DocContent`	M	Base64 byte encoded contents of signed document in appropriate mime type (XML, PDF, etc.). Documents that have strict time bound information should use timestamped digital signatures. This is optional for other documents. The issuer must ensure the safety of digital signature a compromise would result in publishing invalid certificates and as IT Act the owner of the certificate is responsible for such events. If issuers are re-issuing the documents for content alteration or revocations can use pushUri API to notify the locker.

**** END ***