Draft Indian Standard

Smart Cities - Data Exchange Framework: Part 2 Specifications

(First Revision)

BUREAU OF INDIAN STANDARDS

MANAK BHAVAN, 9 BAHADUR SHAH ZAFAR MARG

NEW DELHI 110002

May 2019

CONTENTS

FOREWORD

This is a First Revision of the Part 2 (Specifications) draft of the standard reference architecture for data exchange, under review at the Bureau of Indian Standards, Smart Infrastructure Sectional Committee LITD28.

This standard has been requested by the Ministry of Housing Affairs, Government of India.

1. INTRODUCTION

The next phase of smart cities implementations will leverage **data empowerment**, in order to harness the maximum value from the enormous data cities generate. The current smart city implementations are unable to satisfy this need efficiently, due to the proprietary and ad-hoc nature of the interfaces and their implementations. Hence it is difficult to develop next generation AI/ML based applications for providing new solutions and services at scale, in the current framework. The Data Exchange Framework as discussed in this document aims to address this gap, **by creating a reference architecture (part 1) and interface specifications (part 2)** for interconnecting various IT systems of different government departments as well as external organizations**.**

IUDX consists of set of services that enables access to data resources from one of more of hosting resource servers. IUDX subsystems and interfaces are as follows:

- **Catalogue Server :** A catalogue server hosts different types of meta-information for the data resources in IUDX. The interfaces exposed by the catalogue server are as follows:
    - Search / Discovery (D) - Allows applications to search and discover items.
    - Management (M) - Allows provider applications to manage the item metadata.
- **Authorization Server :** An authorization server serves the policies and handles consent requests using the following interfaces:
    - Authorization (A) - Allows Resource Server to validate the authorisation token of the application.
    - Consent (C) - Allows provider application to define consent policies.
    - Identity (I) - Allows provider identity to be validated.
- **Resource Server :** A resource server serves the data to applications using the following interfaces:
    - Resource Access (R) - Allows applications to access data.
    - Resource Management (RM) - Allows Resource Server to set and control the resource access policies.
    - Service API - Allows resource server to offer value-added high-level services by internally interacting with one or more resource and other APIs so that the application developers can build useful applications with relative ease.

Standardized APIs help build robust application ecosystems that not only improves development cycle times but also leads to improvement in reusability and extensibility of the developed applications. With this objective, the data exchange framework standard defines APIs and ‘Verbs’ for interactions with each interface. Each ‘verb’ has a restricted set of operations that will act distinctly as per the Data Exchange interface it is operated on.

2. SCOPE

This document describes the specifications for the data exchange framework.

The reference architecture is described in part 1.

3. REFERENCES

3.1. Normative References

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content". Available at https://tools.ietf.org/html/rfc7231.

[2] IETF RFC 7232: "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests". Available at https://tools.ietf.org/html/rfc7232.

[3] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax". Available at https://tools.ietf.org/html/rfc3986.

[4] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format". Available at https://tools.ietf.org/html/rfc8259.

[5] IETF RFC 8288: "Web Linking". Available at https://tools.ietf.org/html/rfc8288.

[6] IETF RFC 7946: "The GeoJSON Format". Available at https://tools.ietf.org/html/rfc7946.

[7] IETF RFC 8141: "Uniform Resource Names (URNs)". Available at https://tools.ietf.org/html/rfc8141.

[8] Open Geospatial Consortium Inc. OGC 06-103r4: "OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture". Available at https://portal.opengeospatial.org/files/?artifact_id=25355.

[9] UN/CEFACT Common Codes for specifying the unit of measurement. Available at http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev9e_2014.xls.

[10] IETF RFC 7396: "JSON Merge Patch". Available at https://tools.ietf.org/html/rfc7396.

[11] ISO 8601: 2004: "Data elements and interchange formats -- Information interchange -- Representation of dates and times". Available at http://www.iso.org/iso/catalogue_detail?csnumber=40874.

[12] IETF RFC 2818: "HTTP Over TLS". Available at https://tools.ietf.org/html/rfc2818.

[13] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2". Available at https://tools.ietf.org/html/rfc5246.

[14] IANA Registry of Link Relation Types. Available at https://www.iana.org/assignments/link-relations/.

[15] ISO/IEC 29100:2011(en) Information technology — Security techniques — Privacy framework. Available at https://www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en

[16] The OAuth 2.0 Authorization Framework. Available at https://tools.ietf.org/html/rfc6749

[17] OAuth 2.0 Token Revocation. Available at https://tools.ietf.org/html/rfc7009

[18] MQTT 5.0, OASIS Standard. Available at https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[19] ISO/IEC 19464: Information technology — Advanced Message Queuing Protocol (AMQP) v1.0 specification. Available at https://standards.iso.org/ittf/PubliclyAvailableStandards/c064955_ISO_IEC_19464_2014.zip

[20] IETF RFC 6455, The WebSocket Protocol. Available at http://www.ietf.org/rfc/rfc6455.txt

[21] IETF RFC 2326, Real Time Streaming Protocol (RTSP). Available at http://www.ietf.org/rfc/rfc2326.txt

[22] ISO 19119:2016 Geographic Information - Services. Available at https://www.iso.org/standard/59221.html

[23] [i.25] IS 18XXXX-X (PART 1):2019 Smart Cities - Data Exchange Framework: Part 1 Reference Architecture ***(under review)***

[24] OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding. Available at http://docs.opengeospatial.org/is/12-176r7/12-176r7.html

[25] IUDX Certificate Authority running at https://ca.iudx.org.in

[26] IUDX Authorization Server running at https://auth.iudx.org.in

3.2. Informative References

[i.1] The Personal Data Protection Bill 2018, Govt. of India, http://meity.gov.in/writereaddata/files/Personal_Data_Protection_Bill,2018.pdf

[i.2] Electronic Consent Framework, Technical Specs v1.1, http://dla.gov.in/sites/default/files/pdf/MeitY-Consent-Tech-Framework%20v1.1.pdf

[i.3] National Data Sharing and Accessibility Policy 2012, Govt. of India, https://data.gov.in/sites/default/files/NDSAP.pdf

[i.4] Account Aggregator Technical Standards, Version 1.2, Reserve Bank Information Technology Pvt. Ltd., https://api.rebit.org.in/group

[i.5] Policy on Open Programming Interfaces of Govt. of India, 2015. http://meity.gov.in/writereaddata/files/Open_APIs_19May2015.pdf

[i.6] User-Managed Access (UMA) 2.0, https://docs.kantarainitiative.org/uma/ed/uma-core-2.0-08.html

[i.7] User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization. Available at https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

[i.8] Federated Authorization for User-Managed Access (UMA) 2.0. Available at https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html

[i.9] ETSI GS CIM 009 V1.1.1 (2019-01), Context Information Management (CIM); NGSI-LD API. Available at https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

[i.10] ONVIF Network Interface Specifications. Available at https://www.onvif.org/profiles/specifications/

[i.11] HTTP Live Streaming. Available at https://tools.ietf.org/html/draft-pantos-http-live-streaming-23

[i.12] PAS 212:2016 Automatic resource discovery for the internet of things. Specification. Available at https://shop.bsigroup.com/forms/PASs/PAS-212-2016-download/

[i.13] ISO/IEC 27001, Information technology – Security techniques – Information security management systems – Requirements

[i.14] ISO/IEC 27002, Information technology – Security techniques – Code of practice for thisinformation security controls

[i.15] ISO/IEC 27017, Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services

[i.16] ISO/IEC 27018, Information technology – Security techniques – Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors

[i.17] ISO/IEC 27031, Information technology – Security techniques – Guidelines for information and communication technology readiness for business continuity

[i.18] ISO/IEC 27033 (all parts), Information technology – Security techniques – Network security

[i.19] ISO/IEC 27034 (all parts), Information technology – Security techniques – Application security

[i.20] ISO/IEC 27035 (all parts), Information technology – Security techniques – Information security incident management

[i.21] ISO/IEC 27040, Information technology – Security techniques – Storage security ISO/IEC 29100, Information technology – Security techniques – Privacy framework

[i.22] ISO/IEC29101, Information technology – Security techniques – Privacy architecture framework

[i.23] ISO/IEC 29134:2017, Information technology – Security techniques – Guidelines for privacy impact assessment

[i.24] ISO/IEC 29151, Information technology – Security techniques – Code of practice for personally identifiable information protection

4. TERMINOLOGY AND DEFINITIONS

4.1. Conventions used in the Document

- The key terminologies use `consolas` font type and `dark cornflower blue 3` color encoding.
- The `lowerCamelCase` is used in attribute naming. For nouns, `UpperCamelCase` is used.

Table 3-1: Terminology

Term	Explanation
`Provider`	**Legal Entity**: Human (possibly delegated by an Organization), Organization or an organizational role that has responsibility to provide authorisation to use resources.
`Resource Server`	**Service**: Serves resources to authorized Apps/Consumers.
`Consumer`	**Legal Entity**: Human or Organization or an organizational Role that consumes a resource via a web or mobile `App`.
`App`	**Application:** Software (like a mobile app, web app, device app or server app), that uses resources to provide a service or experience to the `Consumer`.
`ProviderApp`	**Application:** An `App` that enables a `Provider` to manage the meta-data and access control in the data exchange, for the resources they are responsible for.
`Data Exchange Framework`	**Service**: Hosts and manages meta-data about resources and manages authorisation for accessing the resources.
`Consent`	`Provider`'s freely given, specific and informed agreement to the accessing and processing of specific resources in their responsibility.
`Consent Artefact`	A machine-readable electronic document that specifies the parameters and scope of data sharing that a `Provider` consents to in any data sharing transaction.

Personally Identifiable Information	Any information that (a) can be used to identify the PII principal to whom such information relates, or (b) is or might be directly or indirectly linked to a PII principal Note 1 to entry: To determine whether a PII principal is identifiable, account should be taken of all the means which can reasonably be used by the privacy stakeholders holding the data, or by any other party, to identify that natural person.
PII Principal	Natural person to whom the personally identifiable information (PII) relates Note 1 to entry: Depending on the jurisdiction and the particular data protection and privacy legislation, the synonym “data subject” can also be used instead of the term “PII principal”.
Authorization Token	A digital entity that is used to present the authorization credentials to the Resource Server.
Catalogue	A registry of meta-data about the resources in the data exchange available for consumption
resource-item	An entry in the Catalogue that describes the meta-information of the resource that is hosted in an associated Resource Server
DX Authorization Service	Authorization Service of the data exchange
DX Catalogue Service	Catalogue Service of the data exchange
DX Connector	Adapter / Connector service in front of a non-IUDX compliant Resource Server
DX Administrator	**Legal Entity**: Responsible for administering, managing and running the data exchange
DX Certificate Authority	**Service:** Certificate Authority service run by the IUDX

Table 3-2: Abbreviations

Abbreviation	**Definition**
DX	Data Exchange
XML	eXtensible Markup Language

JSON	Java Script Object Notation
API	Application Programming Interface
PII	Personally Identifiable Information
CA	Certificate Authority
RS	Resource Server
AS	Authorization Service
TLS	Transport Level Security
CSR	Certificate Signing Request
CCA	Controller of Certifying Authorities

5. Data Exchange API Interfaces

The high level architecture of the data exchange framework is shown in Figure 4-1. More description is available in Part 1 of this standard.

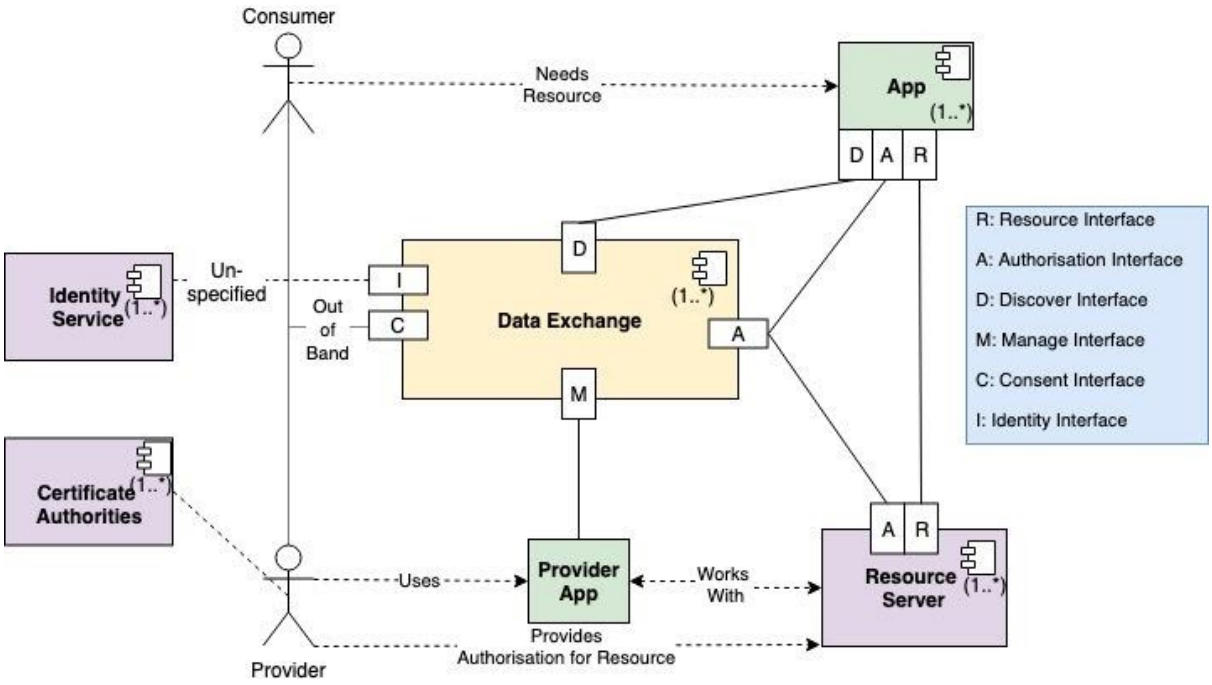

Figure 4-1: High Level Architecture of the Data Exchange Framework

The set of interfaces for this data exchange framework are summarized in the table below:

Table 4-1: Data Exchange Interfaces and APIs

Interface	API Endpoint	Description
Manage	/items /item-groups	Create, update and manage the meta-information of resources in the catalogue.
Discover	/capabilities /search /count	Search the catalogue for capabilities and resources (GET or POST with search queries only).
Resource	/search	Search and retrieve archived data of a resource using complex temporal, spatial and quantitative queries
	/count	Search and retrieve archived data for the count of a resource using complex queries and filters
	/subscriptions	Subscribe to a resource via callback and data stream using complex queries and filters

	/media	Subscribe to a media stream
	/download	Downloads a file
	/metrics	Retrieve the metrics of the APIs used over a time period.
	/reservations	A set of reservation API to manipulate reservation of specific service offered by resource server.
	/groups /reservations /configurations /scheduled-configurations /write /scheduled-write /assets	A set of operational API to operate on the service offered by resource server.
	/operational-status/ /device-status/	A set of status APIs to get the status of services offered by resource server.
Authorization (IUDX)	/auth/v1/acl/set	Create a data sharing policy (post)
	/auth/v1/acl	List all policies (get).
	/auth/v1/token	Get an access token.
	/auth/v1/token/revoke	Revoke a valid access token.
	/auth/v1/token/revoke-all	Revoke all valid access tokens issued to a consumer.
	/auth/v1/token/introspect	Introspect a token
	/auth/v1/audit/tokens	Perform an audit on all issued tokens
	/auth/v1/group/add	Add a consumer to a group
	/auth/v1/group/delete	Delete a consumer from a group
Authorization (UMA 2.0)	/auth/v1/uma/2.0/perm	Request and get authorisation in the form of an authorization code (for OAuth2.0/UMA2.0). https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#rfc.section.4.1
	/auth/v1/uma/2.0/token	Get an access token (for OAuth2.0/UMA2.0)

		https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html#uma-grant-type
	/auth/v1/uma/2.0/introspect	Token introspection point (for OAuth2.0/UMA2.0) https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#token-introspection
	/auth/v1/uma/2.0/revoke	Revoke consent to the auth request for the resource / service page-4 of: https://www.rfc-editor.org/rfc/rfc7009.txt
	/auth/v1/uma/2.0/rreg	Lists all previously registered resource identifiers for this resource owner. https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#list-rreg
	/auth/v1/uma/2.0/rreg/{id}	Reads a previously registered resource description https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#read-rreg

Since APIs may change over time due to changing functional requirements, an API Version Number **shall** be included in every API, as a path parameter. For example, v1, v2 etc.

The remaining document describes the APIs in detail. In particular, we define the Verbs, APIs, Methods, Response and functionalities for different IUDX subsystems.

6. Alignment with OGC

As per the conformance class specification of the OGC Catalogue Services 3.0 Specification - HTTP Protocol Binding [24] standard, there is a minimum set of KVP or XML based requests that should be implemented by a server to conform to an OGC class.

In the Data Exchange APIs, we have adopted the KVP based parameters wherever applicable. The APIs of Catalogue and Resource Interfaces which provide similar features as OGC APIs and are designed with respect to the requirements laid down by the OGC conformance classes. For example, spatial, temporal, attribute and text searches are adopted from the KVP encoding for query constraints and query predicate encoding of OGC Catalogue Services 3.0 Specification - HTTP Protocol Binding [24].

7. IUDX Catalogue Interface

The information resource catalogue contains the meta-data of resources along with auxiliary descriptions, API endpoints, data models and other meta-information like discovery hints, location details, providers etc. A useful analogy is an online shopping catalogue, where a consumer can browse through the available products and then decide to purchase a subscription. The resource catalogue plays a similar role for the consumer applications in the context of smart city data resources.

The catalogue provides interfaces for search (or) discovery and management of meta-information of data resources. The APIs for these functionalities can be constructed using the following sub-modules.

- Base URL
- Item types
- Verbs, Queries and Filters

Authentication and Authorization for the catalogue interface is achieved through the use of client side certificates, issued by the IUDX Certificate Authority [25] This is required only for some of the APIs where there is an explicit mention. Furthermore, the APIs that do require the client side certificates for authentication and authorization, also require the certificates to belong to class 3 of the IUDX certificate classes, as specified in [25]. Any API that requires this kind of authentication and authorization mechanism will mention “IUDX certificate - class 3” in their access mechanism.

7.1. Base URL

The base URL precedes all API endpoints and verbs. It consists of the interface name followed by the API version. The catalogue interface has the following base URL

End-Point	https://{ip:port}/catalogue/{version}
Example	https://pune.iudx.org.in/catalogue/v1

7.2. APIs, Queries and Filters

7.2.1. Capabilities

The Capabilities end point provides capabilities of the server as per the OGC requirement for the capability document [24].
OPERATION : Get ‘capabilities’ of the catalogue

End-Point	/capabilities
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/capabilities
Response	200 OK Document of the capability of server

7.2.2. Items

Any entry in the catalogue is an item. An item can be of multiple item-types as mentioned in the IUDX Reference Architecture document. The list of different itemTypes (referred to as 'item-types' in the context of APIs) and details for each can be found in Section 3.4.1 of the IUDX Reference Architecture document [23] [i.25].

7.2.2.1. Create

Create or Register an item in catalogue.

OPERATION : Creates an item of type item-type in catalogue

End-Point	/items/{item-type}
Method	POST
Status Code	201 Created An application/JSON response of item-id
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/items/resource-item
Request Body	{item_1}
Response	201 Created content-type: application/json

	{ “id” : “80c0b2e2-9ed6-4739-8a09-c62addce5de3” }

7.2.2.2. Retrieve

Read or Retrieve an item using an item-id in catalogue.

OPERATION : Retrieves an item of type item-type in catalogue

End-Point	/items/{id}
Method	GET
Status Code	200 OK An application/JSON response of item-id
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/items/80c0b2e2-9ed6-4739-8a09-c62addce5de3
Response	200 OK content-type: application/json { item }

7.2.2.3. Update

Update an existing item or a group of items of a particular item-type in the catalogue.

OPERATION : Updates an item of type item-type in catalogue

End-Point	/items/{id}
Method	PUT, PATCH
Status Code	200 OK An application/JSON response of update
Example (An example HTTP PUT method)	
Request	https://pune.iudx.org.in/catalogue/v1/items/80c0b2e2-9ed6-4739-8a09-c62addce5de3

Request Body	{item_1}
Response	200 OK

A PUT operation expects the entire item to be presented in the request body. The catalogue replaces the existing item with the latest one.
A PATCH operation expects the fields to be updated in the request body and will only update the respective fields. If the field to be updated is inside a nested-object then the entire nested-object should be presented.

7.2.2.4. Remove

Delete a registered item in catalogue.

OPERATION : Deletes an item of type item-type in catalogue

End-Point	/items/{id}
Method	DELETE
Status Code	204 No Content
Example (An example HTTP DELETE method)	
Request	https://pune.iudx.org.in/catalogue/v1/items/80c0b2e2-9ed6-4739-8a09-c62addce5de3
Response	204 No Content

7.2.3. Item Groups

Catalogue provides an ‘item-group’ API in order to create, update or remove items in bulk (groups). This API can be used by providers to logically tie together a set of resource-items in the catalogue to perform operations like update, delete etc. on all of them at once. An ‘item-group-id’ will be used to identify resource items in the group.

7.2.3.1. Create

OPERATION : Creates group of items of item-type ‘resource-item’ in catalogue

NOTE : *While creating a group, if the provider does not supply a custom group-id, a UUID will be generated by the catalogue for identification of the group*

NOTE: *catalogue-group operations are only applicable to items of type ‘resource-item’*

End-Point	/item-groups/{item-group-id}
Method	POST
Status Code	201 Created
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/item-groups/streetlights
Request Body	[{item_1}, {item_2}, {item_3}, …. , {item_N}]
Response	201 Created content-type: application/json [{ "id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }, { "id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }, { "id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }]

7.2.3.2. Retrieve

OPERATION : Retrieves a group of items of item-type ‘resource-item’ in catalogue

End-Point	/item-groups/{item-group-id}
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/item-groups/streetlights
Response	200 OK content-type: application/json [{

	"id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }, { "id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }, { "id": "80c0b2e2-9ed6-4739-8a09-c62addce5de3", "provider": "PSCDCL" }]

7.2.3.3. Update

OPERATION : Updates group of items of type resource-item in catalogue

End-Point	/item-groups/{item-group-id}
Method	PUT, PATCH
Status Code	204 No Content
Example (An example HTTP PUT method)	
Request	https://pune.iudx.org.in/catalogue/v1/item-groups/streetlights
Request Body	{description: updated description}
Response	204 No Content

7.2.3.4. Remove

OPERATION : Deletes group of items of type resource-item in catalogue

End-Point	/item-groups/{item-group-id}
Method	DELETE
Status Code	204 No Content
Example (An example HTTP DELETE method)	
Request	https://pune.iudx.org.in/catalogue/v1/item-groups/streetlights

Response	204 No Content

7.2.4. Search

A search operation allows applications to discover items in the catalogue using complex queries and filters.

VERB : search
OPERATION : Search items in the catalogue

End-Point	/search
Method	GET, POST
Status Code	200 OK
Response	200 OK content-type: application/json [{ item_1 }, { item_2 }, { item_3 } … { item_n }]

Catalogue allows ‘attribute’, ‘text’ and ‘geo’ search functionalities. This should be used with the ‘search-type’ query parameter to perform a specific type of search.

- Attribute search can find items that match a specific ‘attribute’ query. For example, to find items with ‘tags’=’water’, where ‘tags’ is a known attribute. This is typically expected to be used when certain attributes are known to be present in items.
- Text search can find items that match a given string in some part of a catalogue item.
- Geo search can find items that is within a given geographical boundary.

7.2.4.1. Item Types

OPERATION : Lists all the ‘item-types’ in the catalogue

End-Point	/search?type=item-types
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/catalogue/search?type=item-types

Response	200 OK content-type: application/json [{ item-types : [{item-type-1}, {item-type-2}, {item-type-3}, …. item-type-n}]

OPERATION : Lists all the items of ‘item-type’ resource-server in the catalogue

End-Point	/search?item-type={item-types}
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/catalogue/search?item-type=resource-server
Response	200 OK content-type: application/json [{ resource-servers : [{rs-1}, {rs-2}, {rs-3}, …. rs-n}]

7.2.4.2. Attribute

An attribute search allows applications to perform search on items for an exact match on attributes.

Query Parameter	Query Value	Description
attribute-name	key	Can be any key associated with an item in the catalogue.
attribute-value	[value]	Can be any value associated with an item corresponding to a key in the catalogue. Note that ‘attribute-value’ can be a multi-valued array.

OPERATION: Searches the catalogue for the given attribute query. In case the ‘attribute-value’ is an array the query returns all items with ‘attribute-name’ matching ‘ANY’ of the values specified in the ‘attribute-value’ array.

End-Point	/search

Query Parameters	attribute-name=key&attribute-value=(val_1,..,val_N)
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?attribute-name=tags&attribute-value=(pollution,flood)
Response	200 OK content-type: application/json [{ item_1 }, { item_2 }, { item_3 } … { item_n }]

7.2.4.3. Text

A text search query allows one to search through the catalogue items for a given text string.

Query Parameter	Query Value	Description
q	“text to search”	Can be any free form text query to search the catalogue.

OPERATION: Searches the catalogue for the given text query.

NOTE: *The parts of a catalogue item on which a text search applies is left as an implementation choice. For example, the text string search may apply to text contained in the attribute ‘itemDescription’ or it may apply to the whole item by treating the item as one single string.*

End-Point	/search
Query Parameters	q=”text to search”
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?q=”environment sensors”
Response	200 OK

	content-type: application/json [{ item_1 }, { item_2 }, { item_3 } … { item_n }]

7.2.4.4. Spatial

A geo search allows one to search through catalogue items within a given geographical boundary. This helps in restricting the search query to items belonging to a certain geo-spatial boundary. The search queries and parameters are as per the KVP encoding for query constraints as per the OGC Catalogue Services 3.0 Specification - HTTP Protocol Binding [24].

Search Type	Query Parameter	Query Value	Description
Proximity search	lat	lat-value	Latitude expressed in WGS84.
	lon	lon-value	Longitude expression in WGS84.
	radius	radius-in-meters	The radius of the circle (in meters) centred on the specified location. Points which fall into this circle are considered to be matches.
Arbitrary geometry search	geometry	polygon	A closed polygon whose first and last point must match, thus requiring n + 1 vertices to create an n-sided polygon and a minimum of 4 vertices.
		line-string	An arbitrary line given two or more points.
		multi-point	An array of unconnected points.
		multi-polygon	An array of separate polygons.
		multi-line-string	An array of separate linestrings.
	relation	within	Return all documents whose geo_shape field is fully within the query geometry.
		intersect	Return all documents whose geo_shape field intersects the query geometry.

		contains	Return all documents whose geo_shape field contains the query geometry.
		disjoint	Return all documents whose geo_shape field has nothing in common with the query geometry.
		equals	Returns all documents if two geometries are exactly equal, same coordinates in the same order
		touches	Return all documents whose geo_shape field touches the query geometry.
		overlaps	Return all documents whose geo_shape field overlaps the query geometry.
		crosses	Returns all documents if geo_shape field crosses the query geometry.
		DWithin	Return all documents whose geo_shape field is within a distance mentioned in the query geometry.
	distance	distance-in-meters	For within and beyond geometry relation, distance value is mandatory.
	distance_uom	unit	The unit of measurement of distance, default is meters.
Bounding box search	bbox	Box	A bounding box is expressed to be used as a spatial predicate.

OPERATION: Queries catalogue for items available in a given location

Example : Search and List all items within 2km distance from a given location.

End-Point	/search
Query Parameters	lat=lat-value&lon=lon-value&radius=distance
Method	GET, POST

Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?lat=79.01234&lon=78.33579&radius=2000
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

Example : Search and List all items within 2km distance from a given location.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search
Request Body	{ "lat": 79.01234, "lon": 78.33579, "radius": 2000 }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

Example : Search and List all items within a given polygon.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search

Request Body	{ "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6050,-79.4271))", "relation": "within" }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

7.2.4.5. Search Filters

Search filters will allow combining attribute or text search along with a geo-spatial search.

7.2.4.5.1. Attribute Search Filter

OPERATION: Queries catalogue for items available in a given location. In this case it applies attribute search on top of the geo-spatial search results.

Example : Search and List all items of type resource-item in a given location.

End-Point	/search
Query Parameters	lat=lat-value&lon=lon-value&radius=distance&attribute-name=tags&attribute-value=(values)
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?lat=79.01234&lon=78.33579&radius=2000&attribute-name=tags&attribute-value=(pollution)
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

Example : Search and List all items within a given polygon.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search
Request Body	{ "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6915,-79.4271))", "relation": "within", "attribute-name": "tags", "attribute-value": "(pollution,flood)" }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

7.2.4.5.2. Text Search Filter

OPERATION: Queries catalogue for items available in a given location. In this case it applies text search on top of the geo-spatial search results.

Example : Search and List all environment sensors in a given location.

End-Point	/search
Query Parameters	lat=lat-value&lon=lon-value&radius=distance&q=”text to search”
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?lat=79.01234&lon=78.33579&radius=2000&q=“environment sensors”
Response	200 OK

	content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

Example : Search and List all items within a given polygon with text filter.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search
Request Body	{ "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6915,-79.4271))", "relation": "within", "q": "environment sensors" }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

7.2.4.5.3. Response Filter

Filter acts as the API Presentation layer for applications. Applications can use filters to decide the attributes that needs to be responded for a search query.

Query Parameter	Query Value	Description
attribute-filter	Array of attributes	Can be any key association of an item in the catalogue. Note that ‘attribute-filter’ can be a multi-valued array.

OPERATION: Restricts the response to contain only the attributes requested.

Example : Search and List all environment sensors in a given location.

End-Point	/search
Query Parameters	attribute-filter=(attribute_1,attribute_2,.., attribute_n)
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?lat=79.01234&lon=78.33579&radius=2000&q=”environment sensors”&attribute-filter=(attribute_1,attribute_2,.., attribute_n)
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

Example : Search and List all items within a given polygon with text filter and attribute filter.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search
Request Body	{ "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6915,-79.4271))", "relation": "within", "q": "environment sensors", "attribute-filter": "[id,provider]" }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

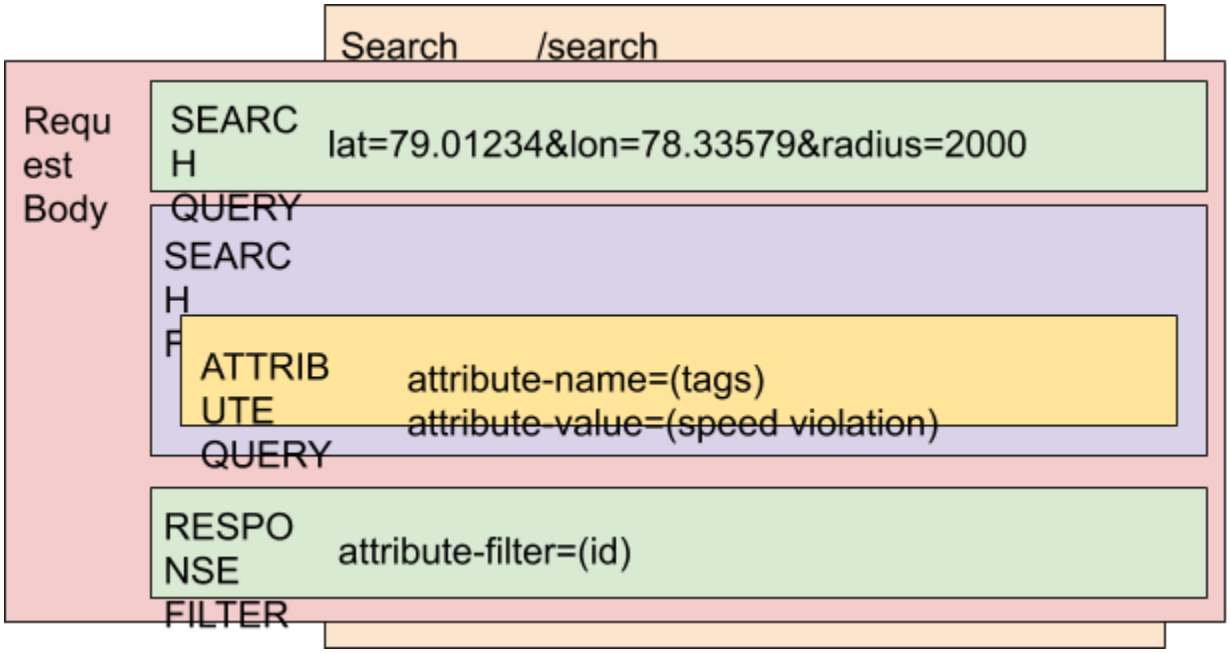

Figure 3 : Search operation in catalogue

Example : Search for resources within a geo-bound that provide speed violation information and present me all the id's.

7.2.4.5.4. Item Type Filter

Search functionalities can also be restricted to a subset of items. By default, the catalogue performs search on all items unless explicitly mentioned. If the user wants to perform search only on resource-items, it can be defined explicitly using the item-type query parameters.

Query Parameter	Query Value	Description
attribute-name	item-type	Specifies the attribute-name on which the value should be applied.
attribute-value	One of {/search?type=item-types}	Attribute value corresponding to which the search shall be applied.

OPERATION: Perform a search on a specific item-type in the catalogue.

Example : Search and List all items of type resource-item

End-Point	/search
Query Parameters	attribute-name=item-type&attribute-value=resource-item
Method	GET, POST
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/catalogue/v1/search?lat=79.01234&lon=78.33579&radius=2000&q=”environment sensors”&attribute-filter=(attribute_1,attribute_2,.., attribute_n)&attribute-value=resource-item
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

7.2.5. Count

A count operator provides a count for the number of hits for a search query. Except ‘attribute-filter’ query parameter, all search query parameters, defined in the above section, are applicable in the count end-point.

OPERATION : Count of all the items in the catalogue

Example : Count all items with pollution tags

End-Point	/count
Method	GET
Status Code	200 OK
Example (An example HTTP GET method) to find the number of pollution sensors	
Request	https://pune.iudx.org.in/catalogue/v1/count?attribute-name=tags&attribute-value=(pollution,flood)
Response	200 OK content-type: application/json { “count” : 100 }

Example : Count all items within a geo-bound

End-Point	/count
Method	GET
Status Code	200 OK
Example (An example HTTP GET method) to find the number of hits in a geo-bound	
Request	https://pune.iudx.org.in/catalogue/v1/count?lat=79.01234&lon=78.33579&radius=2000&q=”environment sensors”
Response	200 OK content-type: application/json { “count” : 55 }

7.2.6. Paging

Paging allows applications to provide a limit and an offset that can be used to paginate through the response.

7.2.6.1. Limit

Allows the application to limit the response size.

7.2.6.2. Offset

Allows the applications to specify the starting item from which the limit shall be applied. Example, start from item 100 and limit to 20 items.

Example : Search and List all items within a given polygon with text filter and attribute filter using limit and offset options.

End-Point	/search?limit=20&offset=100
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/catalogue/v1/search

Request Body	{ "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6915,-79.4271))", "relation": "within", "q": "environment sensors", "attribute-filter": "[id,provider]" }
Response	200 OK content-type: application/json [{ resource-items : [{ item_1 }, { item_2 }, { item_3 } … { item_n }] }]

8. IUDX Resource Server Interface

The resource server provides data access through search, count, subscription APIs. The APIs for these functionalities can be constructed using the following sub-modules.

- Resource Server ID
- Resource Group ID
- Verbs, Queries and Filters

Authentication and Authorization for the resource server interface is through the use of IUDX tokens issued by the IUDX Authorization Server [26]. All the APIs of the resource server interface accept the IUDX auth token using the “auth-token” header. If a token is not provided then the APIs operate only on publicly available data sets or service-offerings. However, when a token is supplied, the resource server interface discerns the scope of the token after contacting the IUDX auth server and performs operations on all those resources instead of only restricting the operation to public resources. More information on the IUDX authorization token can be found at [26]

8.1. Base URL

The base URL precedes all API endpoints and verbs. It consists of the interface name followed by the API version. The resource server interface has the following base URL

Base-URL	https://{ip:port}/resource-server/{resource-server-id}/{version}
Example	https://pune.iudx.org.in/resource-server/pscdcl/v1

8.2. Resource Server

Every resource server is required to register with the catalogue. Upon successful registration, a unique ID, also referred to as resource-server-id, is provided. The base-url for the Resource Server is as follows:

Base-URL	https://{ip:port}/resource-server/{resource-server-id}/{version}/{operation}
Example	https://pune.iudx.org.in/resource-server/pscdcl/v1/search

8.3. Resource Group

It is expected that every ‘resource’ in the ‘resource server’ belongs to a specific type / category / class / device profile which is referred to as ‘resource-group’ in IUDX. This categorization is local to the resource-server and is defined by the Resource Provider. The ‘resources’ belonging to a ‘resource-group’ must share the same data model, share the same access methodologies and request/response objects and must belong to the same ‘resource-server’. Further, a ‘resource’ on a ‘resource-server’ can only belong to one of the ‘resource-groups’. Having such a categorization helps applications to query a subset of resources and also carry out operations on the designated subset. For example: get data from all ‘aqm’ devices that belong to a resource-group ‘aqm-resources’ or get data from all the streetlights from a given vendor which has a specific ‘resource-group’ on a given ‘resource-server’ etc. A resource-group will be identified with a ‘resource-server-id’ which must be unique for a given ‘resource-server’.

End-Point	/resource-server/{resource-server-id}/v1/search
Request Body	{ “resource-group-id” : “<resource-group-id>” }
Example	https://pune.iudx.org.in/resource-server/pscdcl/v1/search Request Body: { “resource-group-id” : “aqm-bosch-climo” }

8.4. APIs, Queries and Filters

8.4.1. Capabilities

The Capabilities end point provides capabilities of the server as per the OGC requirement for the capability document [24].
OPERATION : Get ‘capabilities’ of the catalogue

End-Point	/capabilities
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/capabilities
Response	200 OK Document of the capability of server

8.4.2. Search

A search operation allows querying of archive data of a resource or a group of resources in a resource-group based on temporal, spatial and quantitative parameters.

OPERATION : Queries the Resource Server based on ONE or ALL of time, location and attribute.

8.4.2.1. Temporal

A temporal query allows users to search a resource or a group of resources for a given time query. Each query is associated with a relationship which defines the type of temporal query performed.

OPERATION : Queries a resource or a group of resources using time, end-time and time-relation

Query Parameter	Query Value	Description
time	start/end time as per ISO	Specifies the start-time and end-time.

	8601 format	
TRelation	before	Requests all data before time
	after	Requests all data after time
	TEquals	Requests all data at that time instance
	during	Requests all data between time and end-time

Example : Search for all aqm-bosch-climo-1 data available ‘between’ a given ‘time’ and ‘end-time’

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1” “time” : “2019-04-16T16:06:14,348686568+05:30/2019-04-17T16:06:14,348686568+05:30”, “TRelation” : “during” }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

Example : Search for all aqm-bosch-climo-1 data available ‘before’ a given ‘time’

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1” “time” : “2019-04-16T16:06:14,348686568+05:30”, “TRelation” : “before” }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

8.4.2.2. Spatial

Spatial operations and parameters are as explained in Table [reference].

OPERATION: Queries catalogue for items or specific item-types available in a given location

Example : Search and List all aqm-bosch-climo items within a 2km distance from a given location.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search

Request Body	{ "resource-group-id": "aqm-bosch-climo", "lat": 79.01234, "lon": 78.33579, "radius": 2000 }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

Example : Search and List all aqm-bosch-climo items within a given polygon.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ "resource-group-id": "aqm-bosch-climo", "geometry": "polygon((43.6050,-79.4271,43.6050,-79.3162,43.6915,-79.3162,43.6915,-79.4271))", "relation": "within" }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

8.4.2.3. Attribute

OPERATION : Queries a resource or a group of resources in a resource-group for attribute data as per the operator request.

Query Parameter	Query Value	Description
attribute-name	key	Specifies the name of the attribute on which the operation should be applied.
attribute-value	value	Specifies the value for the attribute with which the operation should be applied.
comparison-operator	operator-name	Specifies the comparison operation that needs to be applied.
logical-operator	operator-name	Specifies the logical operation that needs to be applied for two or more comparison operation.

Operator shall be ONE of the following defined in the table.

Operator Type	Operator Name
Logical Operators	And
	Or
	Not
Comparison Operators	PropertyIsEqualTo
	PropertyIsNotEqualTo
	PropertyIsLessThan
	PropertyIsGreaterThan
	PropertyIsLessThanOrEqualTo
	PropertyIsGreaterThanOrEqualTo

	PropertyIsLike
	PropertyIsBetween

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1”, “attribute” : “temperature”, “value” : “100”, “comparison-operator” : “PropertyIsGreaterThanOrEqualTo” }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

8.4.2.4. Response Filter

Parameter : attribute-filter
OPERATION : Restricts the response to contain only the attributes requested.

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	

Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ “id” : [“pscdcl/aqm-bosch-climo/aqm-bosch-climo-1”, “pscdcl/aqm-bosch-climo/aqm-bosch-climo-2”], “attribute” : “temperature”, “value” : “100”, “comparison-operator” : “PropertyIsGreaterThanOrEqualTo”, “attribute-filter” : [“attribute_1”, “attribute_2”, attribute_3”, … “attribute_n”] }
Response	200 OK content-type: application/json [{ aqm-bosch-climo-1 : [{attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }, {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n } ….. , {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }] }]

8.4.2.5. Options

Options is a query parameter that allows users to obtain specific search results.

Query Parameter	Query Value	Description
options	latest	Gets the latest data of a resource in a resource-group
	status	Gets the status for the resources in a resource-group

8.4.2.5.1. Latest

A latest operation provides the latest known data of resources in a resource-group.

OPTION : latest
OPERATION : Gets the latest data for a resource in a resource-group

End-Point	/search
Method	POST

Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ "id": "pudx-resource-server/flood-sensor" , “item-type”:”resourceServerGroup”, “options” : “latest” }
Response	200 OK content-type: application/json [{item_1}, {item_2}, ….. {item_n}]

8.4.2.5.2. Status

A status operation will provide applications with the status of resources in a resource-group.

OPTION : status
OPERATION : Gets the status for the resources in a resource-group

End-Point	/search
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/search
Request Body	{ "id": "pudx-resource-server/flood-sensor" , “item-type”:”resourceServerGroup”, “options” : “status” }
Response	200 OK

	content-type: application/json { aqm-bosch-climo-1 : live, aqm-bosch-climo-2 : down, ... aqm-bosch-climo-n : maintenance, }

8.4.3. Count

A count operation provides a count for the search. Except attribute-filter, all search functionalities are applicable in the count end-point.

VERB : count

End-Point	/count
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/count
Request Body	{ "resource-group-id": "aqm-bosch-climo", "lat": 79.01234, "lon": 78.33579, "radius": 2000 }
Response	200 OK content-type: application/json { count : 121 }

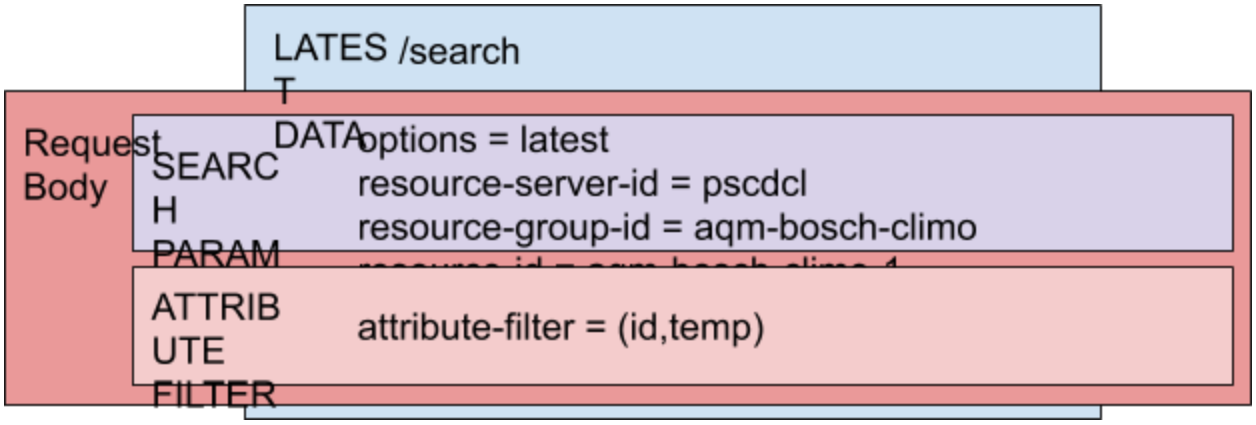

Figure 4 : Filter operations on latest data

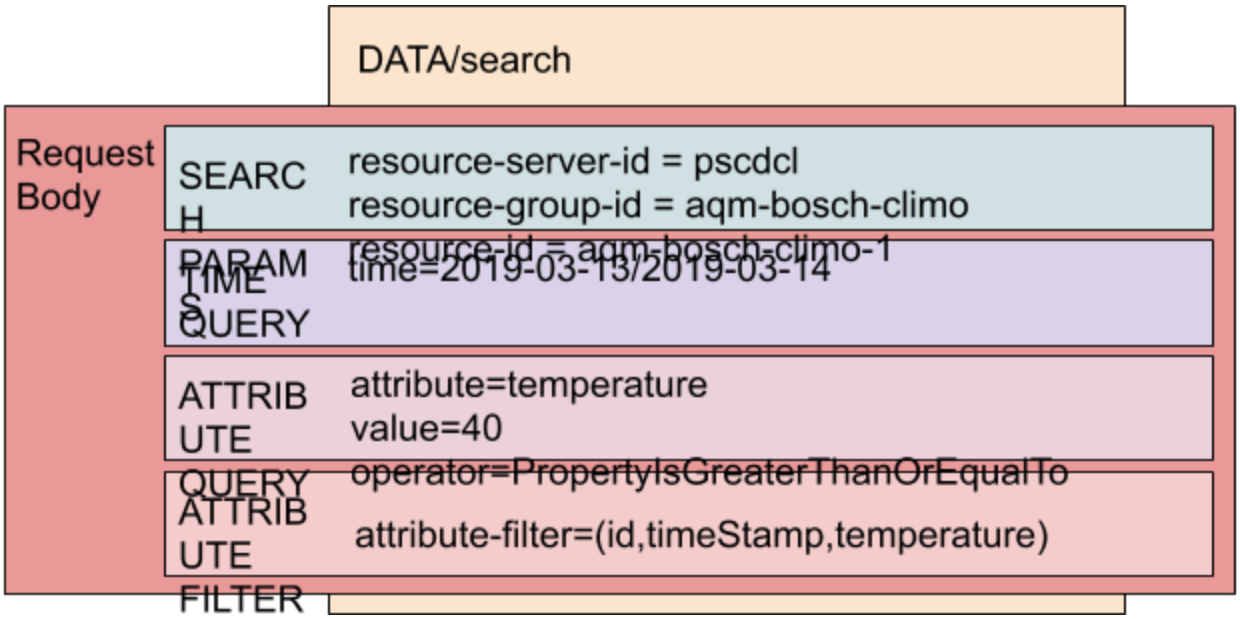

Figure 5 : Query operations on archived data

Example Query : Give me TIME_STAMP, TEMP information of Malleshwaram_CLIMATE_SENSOR where TEMP is PropertyIsGreaterThanOrEqualTo 40 on 14/03/2019

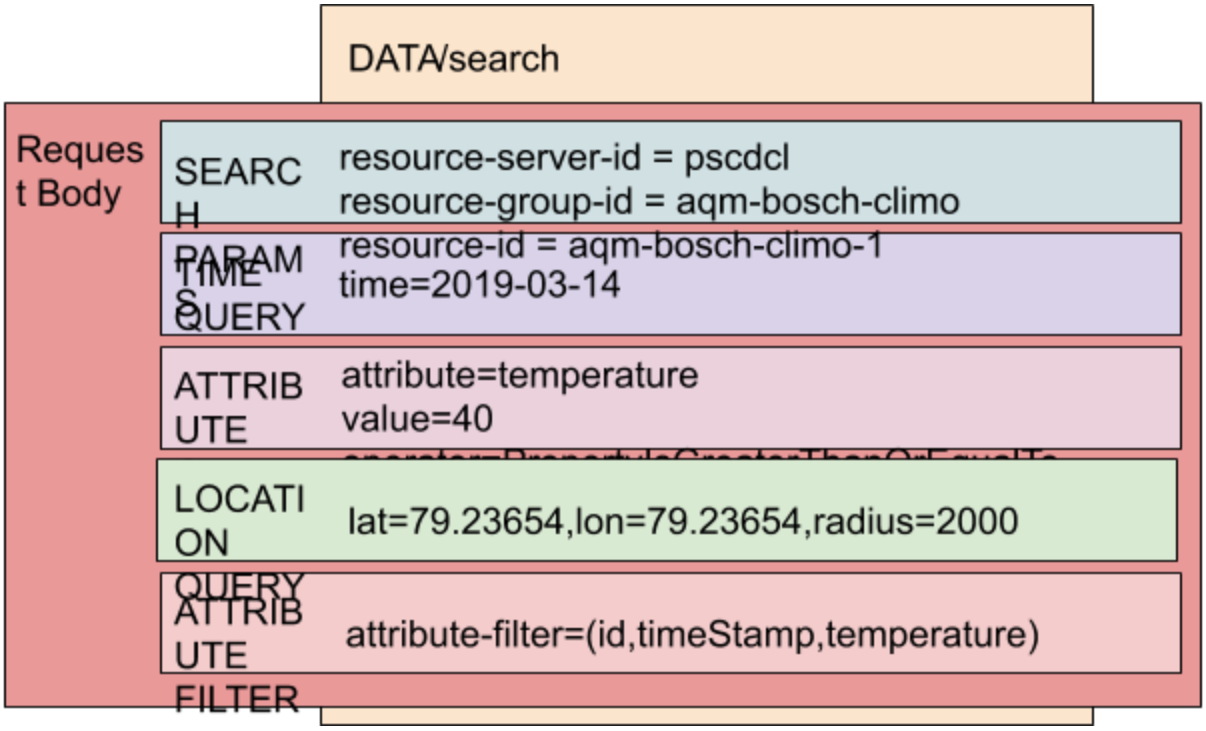

Figure 6 : Location Query operations on archived data

Example Query : Give me NUMBER_PLATE information of all VEHICLES whose speed is GE 80 in Malleshwaram on 14-03-2019

8.4.4. Subscriptions

A subscription operation allows applications to register, update and deregister interest for data access as a stream or as a callback.

A subscription must have either a duration or a limit, where duration is the subscription period (in number of days) and limit specifies the total number of messages subscribed.

8.4.4.1. Types

A subscription request can be made for a stream or for a callback. Subscribing to a stream allows applications to get the data by connecting to the stream URL, whereas with callback subscription the data is posted to the registered callback URL.

8.4.4.1.1. Callback

TYPE : callback
OPERATION : Registers a subscription request for resources through an HTTP callback. The requestor is expected to provide the HTTP endpoint in the request-body for posting the callback.

8.4.4.1.2. Stream

TYPE : stream
OPERATION : Registers a subscription request for resources through AMQP. The requestor will be provided with a streaming URL, to which a subscription call shall be made.

8.4.4.2. Create

OPERATION : Registers a subscription request for resources. The subscription shall contain a single resource or a group of resources in a resource-group, along with (optional) attribute filters.

End-Point	/subscriptions
Method	POST
Status Code	202 Accepted
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions
Request Body	{ “type” : “callback”, “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1”, “duration” : “number-of-days”, “callback_URL” : “https://{my_server_url}:{my_server_port}” }
Response	202 Accepted { “subscription-id” : “e6bf87e7-56be-4487-8d1f-318df6143e62” } Resource Data shall be updated to the requested callback_URL. eg. { aqm-bosch-climo-1 : {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }

End-Point	/subscriptions
Method	POST
Status Code	202 Accepted
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions
Request Body	{ “type” : “callback”, “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1” “limit” : “number-of-messages”, “callback_URL” : “https://{my_server_url}:{my_server_port}” }
Response	202 Accepted { “subscription-id” : “e6bf87e7-56be-4487-8d1f-318df6143e62” } Resource Data shall be updated to the requested callback_URL. eg. { aqm-bosch-climo-1 : {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }

End-Point	/subscriptions
Method	POST
Status Code	202 Accepted
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions
Request Body	{ “type” : “stream”, “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1”, “duration” : “number-of-days” }

Response	202 Accepted { “subscription-id” : “e6bf87e7-56be-4487-8d1f-318df6143e62”, "URL" : "amqp://{userID}:{userPassword}@{serverURL}:{serverPort}/{vhost}", "queue_name" : "queueName" } Resource Data shall be obtained by connecting to the queue using the AMQP stream URL. eg. { aqm-bosch-climo-1 : {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }

8.4.4.3. Update

OPERATION : Updates a subscription request where the request shall contain a single resource or a group of resources in a resource-group.

End-Point	/subscriptions/{subscription-id}
Method	PUT, PATCH
Status Code	202 Accepted
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions/e6bf87e7-56be-4487-8d1f-318df6143e62
Request Body	{ “resource-group-id” : “aqm-bosch-climo”, “resource-id” : “aqm-bosch-climo-1”, “duration” : “30” }
Response	202 Accepted { “subscription-id” : “e6bf87e7-56be-4487-8d1f-318df6143e62”, "URL" : "amqp://{userID}:{userPassword}@{serverURL}:{serverPort}/{vhost}", "queue_name" : "queueName" }

	Resource Data shall be obtained by connecting to the queue using the AMQP stream URL. eg. { aqm-bosch-climo-1 : {attribute_1 : value_1 , attribute_2 : value_2 , ... , attribute_n : value_n }

8.4.4.4. Remove

OPERATION : Deletes a subscription request.

End-Point	/subscriptions/{subscription-id}
Method	DELETE
Status Code	204 No Content
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions/e6bf87e7-56be-4487-8d1f-318df6143e62
Response	204 No Content

8.4.4.5. Retrieve

OPERATION : Retrieve the list of resources subscribed from a resource-server.

End-Point	/subscriptions/{subscription-id}
Method	GET
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/subscriptions/e6bf87e7-56be-4487-8d1f-318df6143e62
Response	202 Accepted { "subscription-id": "e6bf87e7-56be-4487-8d1f-318df6143e62", "items": ["id_1", "id_2", "id_3"] }

8.4.5. Media

Media APIs enable data access from resource servers capable of serving media content. It is expected that that underlying resource-server is capable of streaming various multimedia contents, e.g., video, audio or any other multimedia content etc, to the consuming applications.

8.4.5.1. Types

A media resource can be served in two ways: (1) live and (2) archive. A live media resource will be served as a stream. An archived media resource can be streamed and downloaded.

8.4.5.1.1. Live

TYPE : live
OPERATION : The requestor will be provided with a ‘streaming URL’ to access a live feed. The stream can be accessed by making a subsequent media streaming call, e.g., from a client viewer app, to the provided link.

End-Point	/media
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl-media/v1/media
Request Body	{ “type” : “live”, “resource-group-id” : “pune-sector-1-cameras”, “resource-id” : “pune-sector-1-cam-1” }
Response	200 OK { "playback-url" : "HTTP / RTMP / RTSP playback url" } Media Data shall be obtained by connecting to the server using the HTTP, RTMP, RTSP etc. stream URLs.

The request can also be made for a group of resources, that can be passed as an array of resource-ids in the request body. This is as shown below:

End-Point	/media
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl-media/v1/media
Request Body	{ “type” : “live”, “resource-group-id” : “pune-sector-1-cameras”, “resource-id” : [“pune-sector-1-cam-1”, “pune-sector-1-cam-2”, “pune-sector-1-cam-3”, … “pune-sector-1-cam-n”] }
Response	200 OK { "playback-url" : { “pune-sector-1-cam-1” : "HTTP / RTMP / RTSP playback url", “pune-sector-1-cam-2” : "HTTP / RTMP / RTSP playback url", “pune-sector-1-cam-3” : "HTTP / RTMP / RTSP playback url", “pune-sector-1-cam-4” : "HTTP / RTMP / RTSP playback url" } } Media Data shall be obtained by connecting to the server using the HTTP, RTMP, RTSP etc. stream URLs.

8.4.5.1.2. Archive

8.4.5.1.2.1. Options

8.4.5.1.2.2. Playback

OPERATION : The requester will be provided with a streaming URL for an archived feed, to which a media streaming call shall be made.

End-Point	/media
Method	POST
Status Code	200 OK

Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl-media/v1/media
Request Body	{ “type” : “archive”, “options” : “playback-url”, “resource-group-id” : “pune-sector-1-cameras”, “resource-id” : “pune-sector-1-cam-1” “time” : “2019-04-16T16:06:14,348686568+05:30/2019-04-17T16:06:14,348686568+05:30”, “TRelation” : “during” }
Response	200 OK { "playback-url" : "HTTP / RTMP / RTSP playback url" } Media Data shall be obtained by connecting to the server using the HTTP, RTMP, RTSP etc. stream URLs.

Similar to live, an archive playback request can also be made for a group of resources.

8.4.5.1.2.3. Download

OPERATION : The requestor will be provided with a download URL for an archived feed, to which a media download call shall be made.

End-Point	/media
Method	POST
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl-media/v1/media
Request Body	{ “type” : “archive”, “options” : “download-url”,

	“resource-group-id” : “pune-sector-1-cameras”, “resource-id” : “pune-sector-1-cam-1” “time” : “2019-04-16T16:06:14,348686568+05:30/2019-04-17T16:06:14,348686568+05:30”, “TRelation” : “during” }
Response	200 OK { "download-url" : "HTTP download-url" } Media download shall be done by connecting to the server using the HTTP URLs.

8.4.6. Download

8.4.6.1. Options

Download end-point allows applications to download a file.

OPERATION : Downloads a version of a file from the resource server.

End-Point	/download/{version}/{resource-id}
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/download/1.0.1/filename.ext
Response	200 OK File

8.4.7. Metrics

A metrics operation will provide applications with the usage metrics of resource server.
OPERATION : Gets the API metrics for the resource server

8.4.7.1. Options

An options parameter allows to specify the metrics retrieval requirements. Default is monthly.

8.4.7.1.1. Daily

OPTIONS : daily
OPERATION : Gets the daily API metrics for the resource server between a given time

8.4.7.1.2. Weekly

OPTIONS : weekly
OPERATION : Gets the weekly API metrics for the resource server between a given time

8.4.7.1.3. Monthly

OPTIONS : monthly
OPERATION : Gets the monthly API metrics for the resource server between a given time

8.4.7.2. Time

Time parameter allows to define the time using time, end-time and TRelation using which the report needs to be retrieved.

8.4.7.3. TRelation

Each time query is associated with a relationship which defines the type of temporal query performed.
Example : Get a weekly report between a time and end-time

End-Point	/metrics
Method	GET
Status Code	200 OK
Example (An example HTTP POST method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/metrics
Request Body	{ “options”:”weekly”, “time” : “2019-04-16T16:06:14,348686568+05:30/2019-04-17T16:06:14,348686568+05:30”, “TRelation” : “during” }
Response	200 OK content-type: application/json

	[{ total-no-of-hits : 11223344, no-of-hits-this-week : 112233, most-subscribed-resource: “bosch-aqm-sensor-1”, most-subscribed-resource-group: “bosch-climo” }, ….]

Resource-Types / APIs		**message**	**message -stream**	**Archived message-stream**	**File**	**Table**	**media-stream**
search	*temporal*	N	N	Y	N	O	N
	spatial	N	N	Y	N	O	N
	attribute	N	N	Y	N	O	N
options	*status*	N	N	Y	Y	Y	N
	latest	N	N	Y	N	O	N
	limit	N	N	Y	N	Y	N
count		N	N	Y	N	Y	N
subscription	*callback*	Y	Y	Y	N	N	N
	stream	N	Y	Y	N	N	N
media	*playback-url*	N	N	N	N	N	Y
	download-url	N	N	N	N	N	O
download		N	N	N	Y	N	N

Table : Applicability of APIs on resource-type

Y - Applicable, N - Not Applicable, O - Optional

9. IUDX Service Interface

The service API provides an interface that offers value-added services by internally interacting with one or more resource and other APIs so that the application developers can build useful applications with relative ease. Specifically, the service API deals with the physical resources and services in the smart city. For e.g. an application may need to use resources like Variable Messaging Systems (VMS), Public Announcement Systems and etc. that have been deployed. An app developer may also seek to use the compute hosted by the smart city, to run applications on-prem. Service APIs provide an easy interface to access and use such services.

The resource provider posts the meta information about the offered service into the IUDX catalog server after appropriate authorization. The catalog service allow the authorized consumer to search for available services and retrieve information about supported API's to avail the offered service by that resource, as meta information.

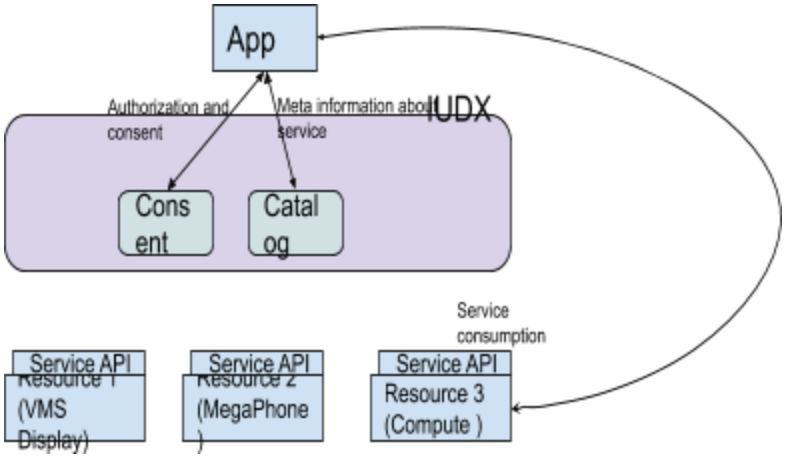

Figure 7 : App consuming service via service API

The meta information thus retrieved helps the consumer to:

1. Discover requisite authorization procedure to access the service
2. Reserve the service, if required
3. Modify the reservation
4. Tear down the reservation
5. Access (read/write) the resources with authorization as applicable
6. Get status of offered service , etc

9.1. List of endpoints in the Service Interface

9.1.1. Groups

The group verb is used to logically tie together a set of resources. This allows the group to be treated as a single logical resource during operations.

9.1.1.1. Create groups

End-Point	/groups
Method	POST
Body	The resources to be included in the group.The resources should be homogeneous in type (e.g. a group of PA systems)
Status Code	201 Created
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/groups/ ["ajb75qol328764", "ajb75qol328765", "ajb75qol328766"]
Response	200 OK Content-type: application/json { "group-id": "qol328764basdn217yh" }

9.1.1.2. Update an existing group

End-Point	/groups/{group-id}
Method	PATCH or PUT
Body	A set of attributes in the group that needs to be updated.
Status Code	200 OK
An example HTTP PUT method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/groups//qol328764basdn217yh { "operation": "add", "resource-list": ["ajb75qol328767"] }
Response	200 OK

9.1.1.3. Delete a group

End-Point	/groups/{group-id}
Method	DELETE
Status Code	204 No Content
An example HTTP DELETE method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/groups/qol328764basdn217yh
Response	204 No Content

9.1.2. Reservations

The APIs in the “reservations” category are used to create, update and delete reservations on resources.

9.1.2.1. Create a reservation

End-Point	/reservations
Method	POST
Body	A set of attributes describing the reservation. Could include startTime, endTime, location, ID, priority, etc. These attributes are specific to the resource being reserved. It also includes {group-id}or{device-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/reservations/ { "group-id": "ajb75qol3", "start-time": "2019 - 04 - 25 T05: 10: 15 + 00: 00", "end-time": "2019 - 06 - 25 T05: 10: 15 + 00: 00", "location": "12.971599,77.594566 "

	}
Response	200 OK Content-type: application/json { "reservation-id": "ajb75qol328764basdn217yha81ma71" }

9.1.2.2. Update an existing reservation

End-Point	/reservations**/**{reservation-id}
Method	PUT/PATCH
Body	The set of attributes in the reservation that needs to be updated.
Status Code	200 OK
An example HTTP PUT method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/reservations/ajb75qol328764basdn217yha81ma71 { "start-time": "2019 - 04 - 27 T05: 10: 15 + 00: 00", "end-time": "2019 - 05 - 25 T05: 10: 15 + 00: 00", "location": "12.971599,77.594566" }
Response	200 OK

9.1.2.3. Delete a reservation

End-Point	/reservations/{reservation-id}
Method	DELETE
Status Code	204 No Content

Example (An example HTTP DELETE method)	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/reservations/ajb75qol328764basdn217yha81ma71
Response	204 No Content

9.1.3. Assets

These APIs can be used to manage assets that need to be uploaded to devices during reservation periods

9.1.3.1. Create Assets

This API can be used to upload assets to the resource server. E.g. a message to a VMS, an audio file to an announcement system.

End-Point	/assets
Method	POST
Body	The content to be displayed on a resource and the templates to display the content. It also includes {user-id}
Status Code	201 Created
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/assets { "user-id": "ajb125", “content": "Weather Forecast" “Content-name”: “Weather” }
Response	201 Created Content-type: application/json { "asset-id": "ajb125/Weather" }

9.1.3.2. Update Assets

End-Point	/assets/{asset-id}
Method	PATCH or PUT
Body	A set of attributes in the content of the resource that need to be updated. It also includes {content-id}
Status Code	200 OK
An example HTTP PUT method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/assets/ajb125%2fWeather { "content-id": "ajb125/Weather", "message": "Bad weather for the next 48 hours" }
Response	200 OK

9.1.3.3. Delete Assets

End-Point	/assets/{asset-id}
Method	DELETE
Body	A set of attributes in the content of the resource that needs to be deleted. It also includes {asset-id}
Status Code	204 No Content
An example HTTP DELETE method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/assets/ajb125%2fWeather

	{ "content-id": "ajb125/Weather" }
Response	204 No Content

9.1.4. Scheduled Configurations

The APIs in the “scheduled configurations” category are used to create configurations of specific resources prior to the commencement of the reservation duration obtained for the device. These configurations are applied into the respective devices if there is no high-priority task running at the time.

End-Point	/scheduled-configurations
Method	POST
Body	A set of attributes describing the configuration. Could include volume, brightness, contrast, etc. These attributes are specific to the resource being configured. It also includes {group-id}or{device-id} along with {reservation-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/scheduled-configurations/ { "reservation-id":"ajb75qol328764basdn217yha81ma71", "volume": 75, "brightness": 80, "contrast": 90 }
Response	200 OK

9.1.5. Scheduled Write

This is used to write data on a resource asynchronously i.e prior to the commencement of the reservation period for the device(s)

End-Point	/scheduled-write
Method	POST
Body	The data to be written to a resource. It also includes {group-id}or{device-id} along with {reservation-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/scheduled-write { "reservation-id": "ajb75qol328764basdn217yha81ma71", "message": "Welcome to Pune smart city" }
Response	200 OK

9.1.6. Channel

The channel API is used to get a confirmation from the resource server that the set of resources to be used are free to be operated on.

9.1.6.1. Open a channel

This API will return a 200 OK if the set of resources are free to be operated on, i.e. no high priority task is running

End-Point	/channels
Method	POST
Body	It includes {reservation-id}or{group-id}or{device-id}
Status Code	200 OK
An example HTTP POST method	

Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/channels { "channel-id": "ajb75qol328764basdn217yha81ma71" }

9.1.6.2. Relinquish a channel

This API relinquishes a channel while destroying any and all configurations made by the user.

End-Point	/channels/{channel-id}
Method	DELETE
Body	Contains channel-id
Status Code	204 No Content
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/channels/ajb75qol328764basdn217yha81ma71
Response	204 No Content

9.1.7. Configurations

The APIs in the “configurations” category are used to create, update or delete configurations of specific resources during the reservation time.

9.1.7.1. Create a configuration

End-Point	/configurations
Method	POST

Body	A set of attributes describing the configuration. Could include volume, brightness, contrast, etc. These attributes are specific to the resource being configured. It also includes {group-id}or{device-id} along with {reservation-id}or{channel-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/configurations { "reservation-id": "ajb75qol328764basdn217yha81ma71", "volume": 75, "brightness": 80, "contrast": 90 }

9.1.7.2. Update an existing configuration

End-Point	/configurations
Method	PATCH/PUT
Body	A set of attributes in the configuration that need to be updated. It also includes {group-id}or{device-id} along with {reservation-id}or{channel-id}
Status Code	200 OK
An example HTTP PUT method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/configurations { "reservation-id": "ajb75qol328764basdn217yha81ma71", "volume": 85 }
Response	200 OK

9.1.7.3. Delete an existing configuration

End-Point	/configurations/

Method	DELETE
Body	It includes {group-id}or{device-id} along with {reservation-id}or{channel-id}
Status Code	204 No Content
An example HTTP DELETE method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/configurations/ { "reservation-id": "ajb75qol328764basdn217yha81ma71" }
Response	204 No Content

9.1.8. Write

This is used to write data, update data and delete data on a resource during the reservation time.

9.1.8.1. Write Data

This API can be used to write data to a resource. E.g. sending a message to a VMS, uploading an audio file

End-Point	/write/
Method	POST
Body	The data to be written to a resource. It also includes {group-id}or{device-id} along with {reservation-id}or{channel-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/write/ { "reservation-id": "ajb75qol328764basdn217yha81ma71", "message": "Welcome to Pune smart city" }

Response	200 OK

9.1.8.2. Update Data

End-Point	/write/
Method	PATCH or PUT
Body	A set of attributes in the data of the resource that need to be updated. It also includes {group-id}or{device-id} along with {reservation-id} or {channel-id}
Status Code	200 OK
An example HTTP PUT method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/write { "reservation-id": "ajb75qol328764basdn217yha81ma71", "message": "Bad weather for the next 48 hours" }
Response	200 OK

9.1.8.3. Delete Data

End-Point	/write/{resource-server-id}
Method	DELETE
Body	A set of attributes in the data of the resource that need to be deleted. It also includes {group-id}or{device-id} along with {reservation-id} or {channel-id}
Status Code	204 No Content
An example HTTP DELETE method	

Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/write { "reservation-id": "ajb75qol328764basdn217yha81ma71" }
Response	204 No Content

9.1.9. Status

This API can be used to get the operational and device status of the resource.

9.1.9.1. Operational Status

End-Point	/operational-status/
Method	POST
Body	It includes the operational parameters whose status user wants to know. It also includes {group-id}or{device-id} along with {reservation-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/operational-status { "device-id": "ajb75qol328767", "parameters": ["reservation-status", "queue-status", "content-status"] }
Response	200 OK Content-type: application/json { "reservation-status": "unreserved", "queue-status": "empty", "Content-status": "Welcome to Pune smart city" }

9.1.9.2. Device Status

End-Point	/device-status/
Method	POST
Body	It includes the technical parameters whose status user wants to know. It also includes {group-id}or{device-id} along with {reservation-id}
Status Code	200 OK
An example HTTP POST method	
Request	https://pune.iudx.org.in/resource-server/pscdcl/v1/device-status { "device-id": "ajb75qol328767", "parameters": ["device-power", "volume"] }
Response	200 OK Content-type: application/json { "device-power": "on", "Volume": 76 }

10. IUDX Security APIs

10.1. Access control list

Access control list (acl) is a list of data sharing policies for a provider.

OPERATION : Create a policy for a provider.

End-Point	/auth/v1/acl/set
Method	POST

Status Code	200 OK
Example	
Request	https://auth.iudx.org.in/auth/v1/acl **Header:** content-type: application/json **Body:** {"policy":"barun@iisc.ac.in can access pune.iudx.org.in/streetlight-1 for 10 days"}
Response	200 OK

OPERATION : List all policies of a provider.

End-Point	/auth/v1/acl
Method	GET
Status Code	200 OK
Example (An example HTTP GET method)	
Request	https://auth.iudx.org.in/auth/v1/acl
Response	200 OK {"policy":"barun@iisc.ac.in can access pune.iudx.org.in/streetlight-1 for 10 days"}

10.2. Access tokens

OPERATION: Get an access token

End-Point	/auth/v1/token

Request	https://auth.iudx.org.in/auth/v1/token content-type: application/json [{ "resource-id": <id>, "api": <api-string>, "methods": [<method-1>, <method-2>, ...] }, ...]
Method	POST
Status Code	201 Created
Response	content-type: application/json { "access_token": <token>, "token_type": "IUDX", "expires_in": <expiry-time-in-seconds> }

OPERATION: Revoke a valid token

End-Point	/auth/v1/revoke
Request	https://auth.iudx.org.in/auth/v1/token/revoke {"tokens": [<list of-tokens-to-be-revoked>]}
Method	POST
Status Code	200 OK
Response	content-type: text/json {"success":true}

OPERATION: Introspect a token.

End-Point	/auth/v1/token/introspect
Request	https://auth.iudx.org.in/auth/v1/introspect **Body:** {"token": <token-to-be-checked>}
Method	POST
Status Code	200 OK
Response	{ "active": true, "username": <username>, "expiry": <token-expiry-time>, "user-certificate-class": <certificate-class-used-by-the-consumer>, "request": [{ "resource-id": <resource-id> "methods": [<method-1>, <method-2>, ..] }] }

Introspect an API.

End-Point	/auth/v1/audit/tokens
Request	https://auth.iudx.org.in/auth/v1/audit/tokens **Body:** {"hours": <number-of-hours-for-which-audit-report-has-to-be-generated>}
Method	POST
Status Code	200 OK
Response	[{ "token-issued-at": <time>,

	"expiry": <time>, "certificate-serial-no": <serial-number-of-certificate-used-to-get-token>, "certificate-fingerprint": <fingerprint-of-certificate-used-to-get-token>, "request": <request-made-in-json>, "introspected": <true/false> }, …]

OPERATION: Add consumer to a group.

End-Point	/auth/v1/group/add
Request	https://auth.iudx.org.in/auth/v1/group/add **Body:** { "consumer": <the name of the consumer> "group": <the name of the group to which the consumer has to be added> "valid-till": <the number of hours for which the group membership is valid> }
Method	POST
Status Code	200 OK
Response	-

OPERATION: Delete consumer from a group.

End-Point	/auth/v1/group/delete
Request	https://auth.iudx.org.in/auth/v1/group/delete **Body:** { "consumer": <the name of the consumer> "group": <the name of the group from which the consumer has to be deleted> }
Method	POST

Status Code	200 OK
Response	-

OPERATION: Delete an entire group.

End-Point	/auth/v1/group/delete
Request	https://auth.iudx.org.in/auth/v1/group/delete **Body:** {"consumer":"*","group": <the name of the group which has to be deleted>}
Method	POST
Status Code	200 OK
Response	-

10.3. UMA 2.0 and RFC compatible APIs

10.3.1. Permissions

Request and get authorisation in the form of an authorization code (for OAuth2.0/UMA2.0).

https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#rfc.section.4.1

10.3.2. Token

Get an access token (for OAuth2.0/UMA2.0)

https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html#uma-grant-type

10.3.3. Introspect

Token introspection point (for OAuth2.0/UMA2.0)

https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#token-introspection

10.3.4. Revoke

Revoke consent to the auth request for the resource / service

page-4 of:
https://www.rfc-editor.org/rfc/rfc7009.txt

10.3.5. Resource registration

Lists all previously registered resource identifiers for this resource owner.

https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html#list-rreg