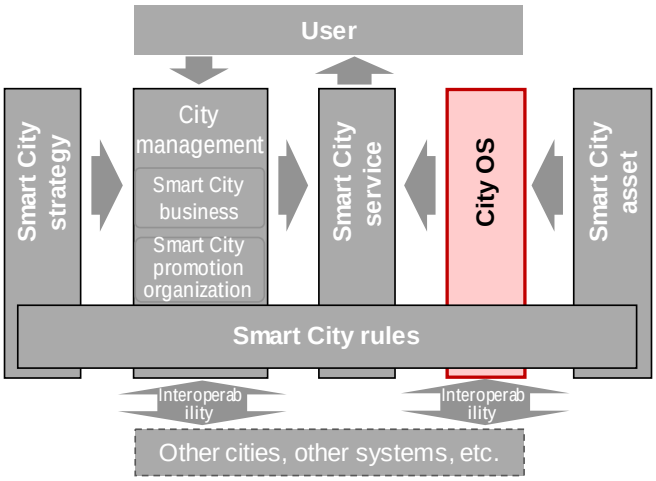
7. City OS

7.1 City OS Overview



As the overview of City OS, the characteristics and structure of City OS are presented here. Figure 7.1-1 shows the method to develop City OS in Smart City Reference Architecture. City OS was designed, first by consolidating the necessary characteristics of City OS in view of the issues in the realization of Smart City in Japan, and then by consulting the concept of Smart City in Japan derived from "Society 5.0" and also by referencing Smart City architectures found in other countries.

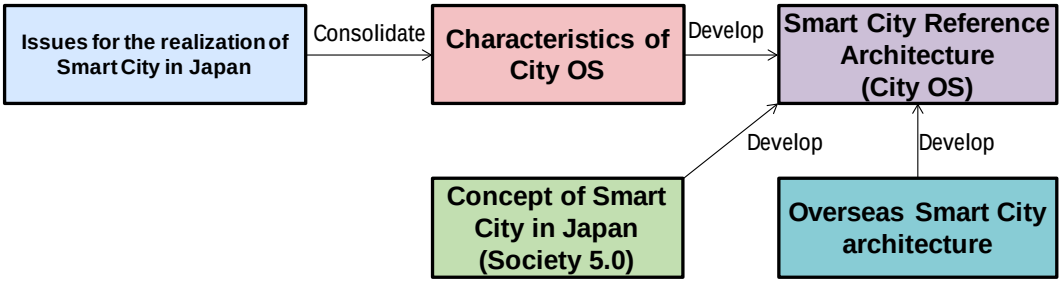


Figure 7.1-1 Development of City OS in Smart City Reference Architecture

7.1.1 Characteristics of City OS

Figure 7.1-2 shows the characteristics of City OS. There are three issues to consider for the realization of Smart City in Japan, namely 1) replication and lateral expansion of services, 2) cross-sectoral utilization of data, and 3) scalability. In reference to 1) many of the systems so far deployed in various organizations and sectors have been customary built, and therefore, the services/solutions cannot be easily replicated to other regions. As regards to 2) from the viewpoint of cross-sectoral data utilization, it is difficult to develop novel services leveraging information available from multiple sectors because conventional services and data have been built and exist in a fragmented way within each sectoral silo. As for 3) scalability, services cannot be easily and continuously improved as

it is costly and time-consuming to expand or upgrade services based on the conventional function-specific systems.

In order to address these issues for the realization of Smart City in Japan, City OS is designed with such characteristics of as 1) interoperability (connect), 2) data exchange (flow), and 3) scalability (future-proof).

1) Interoperability (connect) is the mechanism which allows replication of outcomes and best practices created in a region to another and also allows enables federation of services within and among regions/communities. It is achieved by implementing common functions and standard interfaces, together with a mechanism of opening them to public.

2) Data exchange (flow) is a mechanism to make the flow of data easier within and across regions by making various data existing in sectoral as well as organizational silos accessible as if they exist in a shared and logically-configured way. This necessity calls for a mechanism in which City OS acts as a broker of heterogeneous data (various forms of data within and outside City OS).

3) Scalability (future-proof) is a mechanism with which City-OS can be continuously curated and improved, which refers to an easy upgradability and extendibility of its functions when redefinition of issues or goals of a region takes place or the Smart City Reference Architecture is updated. For such a purpose, it is necessary to build the systems consisting of loosely-coupled functions, thereby enabling the upgrade or expansion only of the necessary components suffices

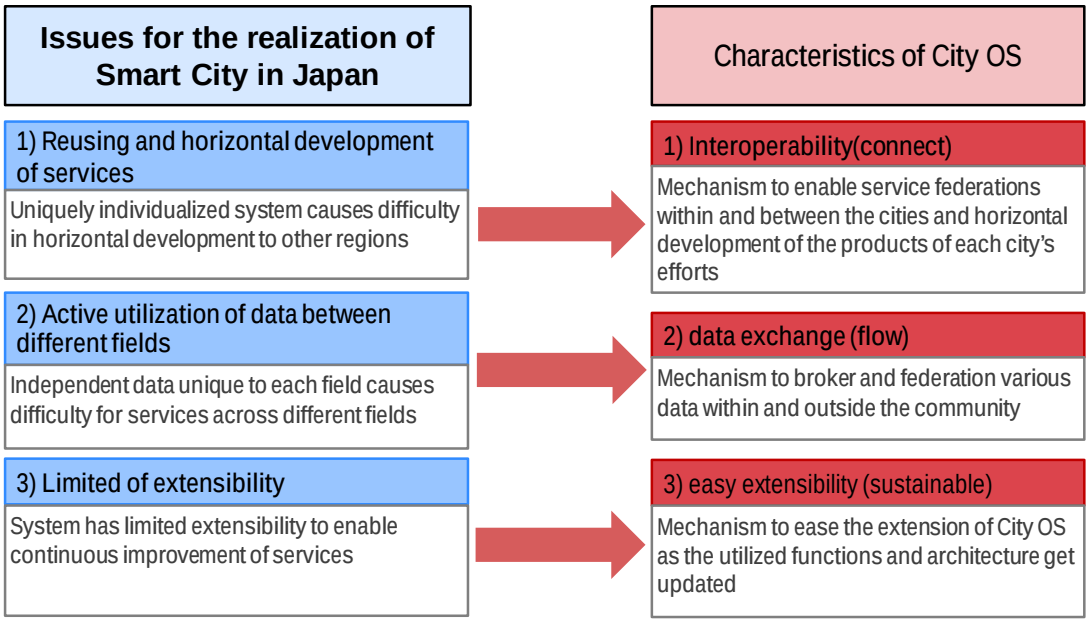


Figure 7.1-2 Characteristics of City OS

7.1.1.1 Interoperability (connect)

There are various ways how regions/regions in Japan will collaborate. Accordingly, as shown in Figure 7.1-3 shows, there should be several options regarding how Smart City services are federated, namely corresponding to a stand-alone city, cooperating multiple cities and distributed multiple cities/areas. In any case, interoperability among City-OS is essential in order to replicate outcomes to other regions.

The interoperability of City OS refers to the situation where APIs and data are provided in a common format/protocol or through some appropriate machine conversion so that federation among various Smart City services and with another City OS can be accomplished.

In order to guarantee interoperability, it is important to implement following two policies for various functions (API, data) provided by City OS so that systems can be interconnected and work together when necessary.

1) Actively adopt APIs, data model, etc. defined by standardization bodies

2) Adopt mechanism to make APIs and Data open to public to allow access by various entities can access

City OS can be used in a common way under any of the inter-regional cooperation model by implementing functional requirements and standardized specifications listed in “7.3.2 API and interface provided by City OS”. Each function (API, data) should be provided in the form of open API and open data (or closed API and closed data accessible only by authorized personnel). Please refer to “7.3.1.3 Interoperability of City OS” for details.

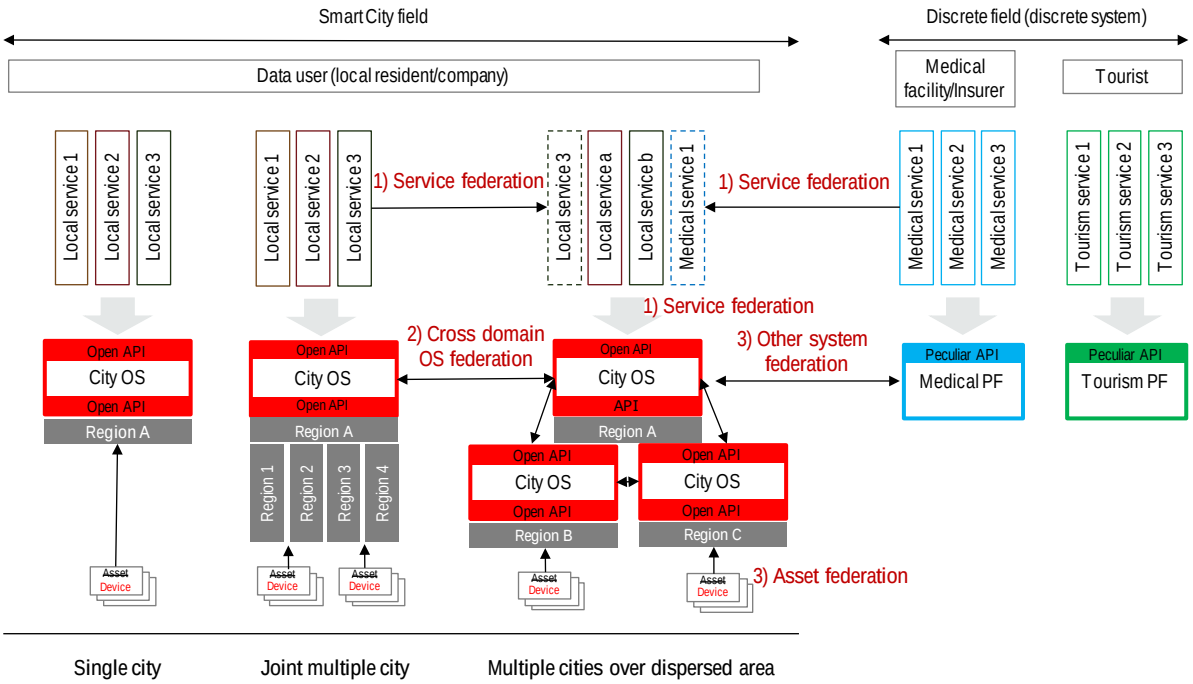


Figure 7.1-3 Smart City service federation enabled via interoperable City OS

Interoperability in City-OS will lead to three types of outcomes, namely 1) service federation, 2) cross domain OS federation, and 3) federation of assets/other systems as shown in Table 7.1-1.

Table 7.1-1 Merits of Interoperability

Reference to diagram	**Merit**
1) Service **federation**	By defining API provided by City OS, a Smart City service can be plugged into the City OS of the region, and further can be replicated to another City OS that is interoperable
2) Cross domain OS **federation**	By federating multiple City OS among regions, thereby enabling exchange of data through interoperable City OS, it will become possible to provide Smart City services beyond regional boundaries and enhance citizen's convenience also by sharing data across regions. Furthermore, creation of new local business or economy is also expected leveraging analysis of region-specific market characteristics.

3) Federation of Assets / Other systems	By sharing various Smart City assets within the region and data held by other systems via City OS, cross-domain services will be enabled going beyond the silo walls of organizations and systems.

7.1.1.2 Data exchange (flow)

Smart City is required to provide cross-domain Smart City services to resolve issues breaking the walls separating domains and organizations. In order to realize that, City OS must be implemented with functions to distribute heterogeneous types of data (various types of data within and outside City OS). This function to enable exchange of heterogeneous data is called “data brokering” or “Broker” and needs to support 1) handling of various types and forms of data, and 2) brokering of data within and outside City OS.

1) Handling of various types and forms of data

As City OS handles various types of data having different characteristics, it is necessary to appropriately manage the data related to the issues to be resolved by the region based on their characteristics.

Table 7.1-2 and Figure 7.1-4 show some examples of data types handled by City OS.

Table 7.1-2 Examples of data types

#	Data type	Description
1	Meta data	Associated data describing the data model (data items and format) and attribute information, etc. of the data body, used for efficient management and search of the data (static data, dynamic data, personal data, etc.) The classification of data includes context data and data catalogues. The data body is managed by searching via metadata and accessing according to the characteristics of each data type.
2	Static data	Infrequently updated data stored for an extended period and referenced. The classification of data includes statistics data, analysis data, historical data, document data, etc. They should often be made available as open data, and it is necessary mainly for municipalities to publish the data they hold as open data in accordance with the rules for handling them, and make efforts to promote its use.
3	Dynamic data	Frequently updated chronologically continuous data generated in real-time. The classification of data includes sensor data, log data, people flow data, streaming data, etc. As they are generated real-time, it is necessary to tag them with time stamps, or manage the log, etc.
4	Geospatial data	Data that contains information identifying the location of a specific point or area in a space (location information). It may contain information associated with various events at the location. The classification of the data includes geospatial information such as topographical maps, aerial photographs, satellite images, etc., and BIM (Building Information Modeling) data, CIM (Construction Information Modeling) data, etc. associated with buildings and civil engineering structures.
5	Personal data	In addition to generally-conceived personal data, they also include those which may not intuitively be classified as personal data, in other words, a wide range of data regarded to be related to an individual such as the individual's attributes information, logs of movement, activities, and purchase, the data acquired from wearable devices, or processed information thereof. The classification of the data includes special-care required personal information, personal information, and anonymized information. In compliance with the rules for the handling of personal data, privacy protection and advanced security measures are required.

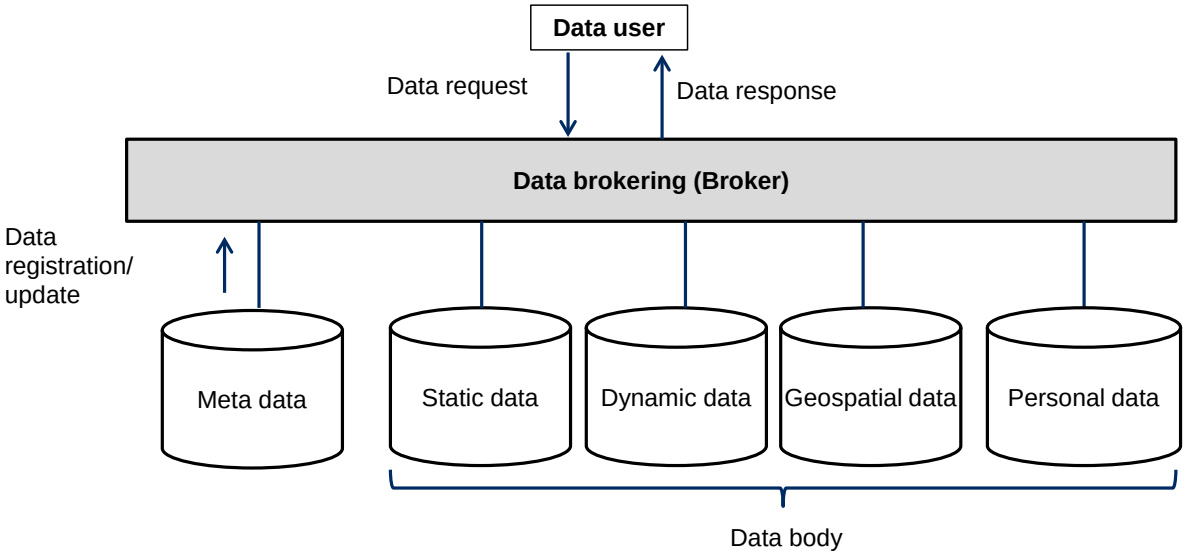


Figure 7.1-4 Data types and data broker of City OS

2) Data brokering within and outside City OS

Data broker is required to work on the data both within and outside City OS. The brokering scheme is classified into two categories, namely for accumulated data and for distributed data. However, it is possible for users to access data without knowing which.

Table 7.1-3 Classification of data broker methods

Broker method	Description
Data accumulation method	It registers and accumulates data on City OS and centrally registers and manages.
Data exchange method	City OS does not register nor accumulate data but only manages the location and availability of the distributed data. Utilizing the location and availability information, City OS brokers the data in response to the data access request from users.

Figure 7.1-5 Data brokering options according to the federation scheme of City OS.

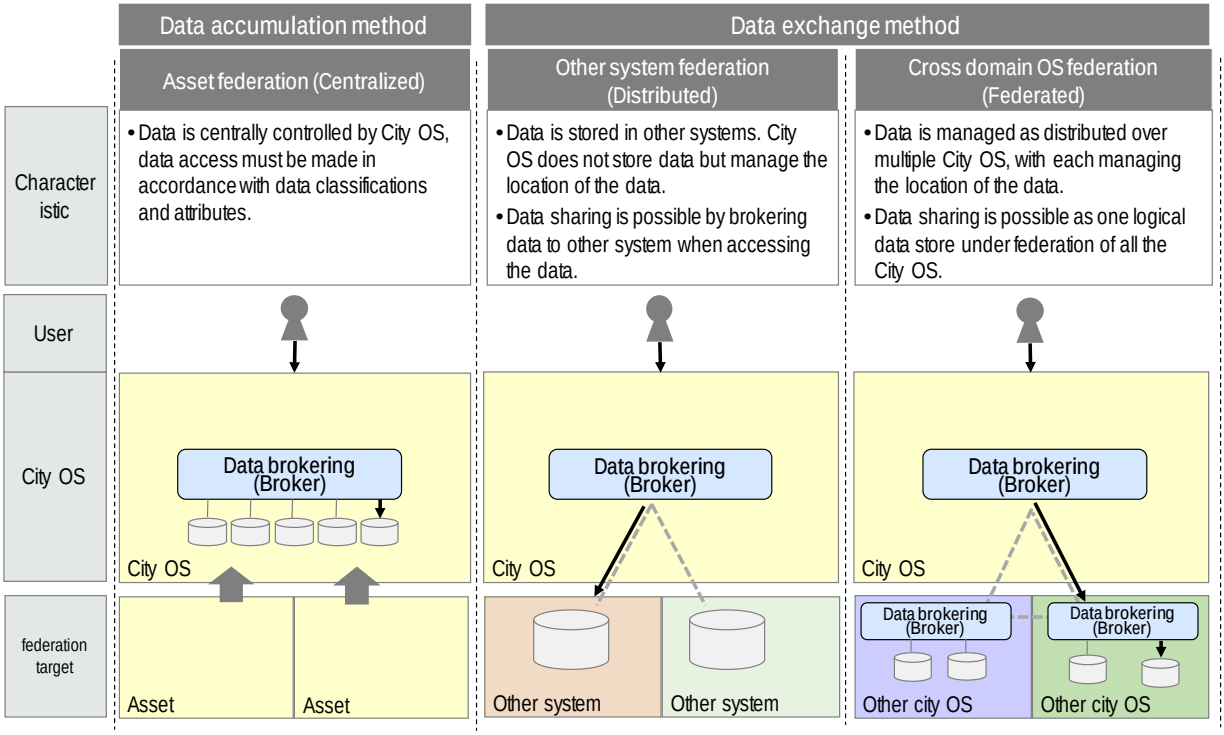


Figure 7.1-5 Examples of data broker by federation method of City OS

Virtualization of real world in the form of city digital twin is an example of the benefit resulting from the aggregation and utilization of various types of distributed data by way of these data brokering. City digital twin can be defined as the digital environment which is comprised of activities and environment information of the city centered around spatial information of the city and real-time information obtained via IoT, and visualizes the status of the city in real time in wide range of Smart City service domains such as mobility, tourism, disaster prevention, infrastructure maintenance and management, environment, energy, innovation initiatives, etc. to enable advanced information processing in the city space such as energy, people-flow, simulation of traffic flow, holistic optimization, prediction making, data-driven decision-making assistance, etc. (Reference: “Appendix D Trends in digitalization of cities”)

Based on the latest trends in the movement towards city digital twin, its components of spatial information and activities & environment information of the city can be categorized as shown in Figure 7.1-6.

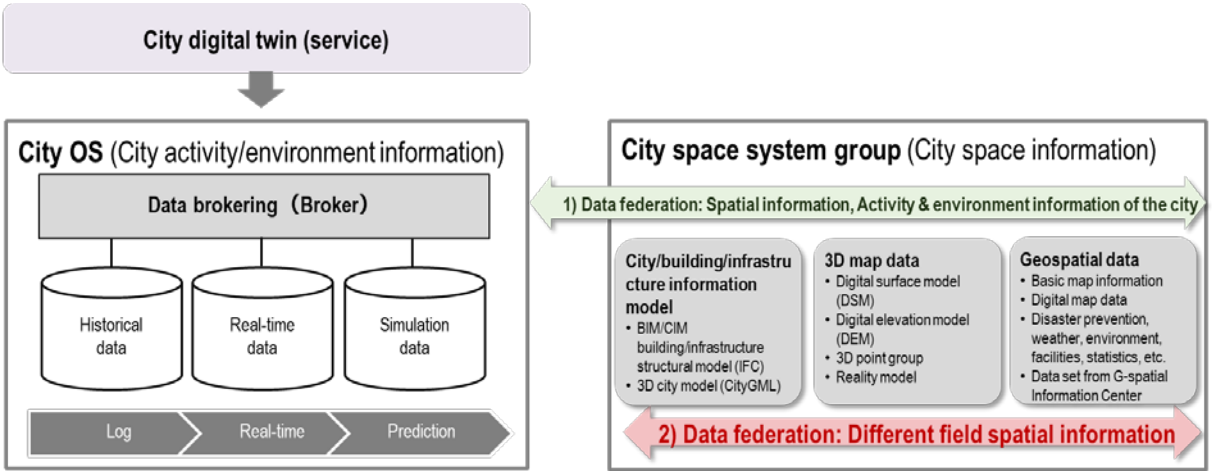


Figure 7.1-6 City digital twin

- City OS (Activity & environment information of the city)
  - Real-time (dynamic data): IoT and social media data with location information, etc.
  - Historical (static data): Data with location and time information added to real-time data, etc.
  - Prediction (static data): Simulated results using data for people flow, traffic flow, wind/light/sound environment, natural disaster, etc.
- City space systems (Spatial information of the city)
  - City/building/infrastructure information model: 3D information model including the attribute information of city space, building, and infrastructure. CityGML[32], IFC (Industry Foundation Classes)[33], etc. are being utilized as data models.
  - 3D map data: Reality models based mainly on Digital surface model, Digital topography model, 3D point group, and 3D geometrical forms.
  - Geospatial data: Basic map data, Digital map data, and various geospatial data in the fields of disaster prevention, weather, facilities, statistics. Data sets accessible from G-spatial Information Center[34] fall into this category.

7.1.1.3 Scalability (future-proof)

In Smart City, it is necessary to continually add and update functions in accordance with the issues to be resolved in the region and the future goals they should aim at. It is thus necessary implement a mechanism that allows flexible reconfiguration of functionalities, for instance, by leveraging building-block approach.

---

[32] CityGML https://www.ogc.org/standards/citygml

[33] IFC https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/

[34] G-spatial Information Center https://www.geospatial.jp/gp_front/

The building block method is a method to build a loosely-coupled system by selecting and adopting necessary functions from functional groups. A building block refers to a group of functions that are integrated to some extent. To enable interoperability in City OS, it is desirable to construct by way of microservices for which communication between building blocks are consolidated or published in the form of APIs. As a result, it becomes possible to make updates in a certain building block with giving minimal impact to other building blocks and services, and hence improves overall maintainability. Furthermore, it is possible to start with a small size in the beginning and gradually extend the functions in view of the issues to be resolved in the region and the future goals they should aim at. In the future, it is desirable that the addition of federating Smart City services and Smart City assets, as well as the changes in the various functions provided, can be easily reconfigured by visual operations.

Furthermore, City OS is expected to maintain transparency by adopting open-source as much as possible without relying on a particular vendor. Under such circumstances, by promoting open access to various APIs between building blocks and thereby ensuring interoperability, it becomes possible for various organizations to participate as an integral part of City OS, and eventually leads to further development of City OS and the region. However, with the expansion of City OS as a result of the region growth, new issues may emerge when the system becomes too complex and starts to look like a black-box. It is therefore desirable to examine the functional designs and operations in order to ensure the systems to be easily scalable and also to stay interoperable

Figure 7.1-7 shows an example of construction by building block method.

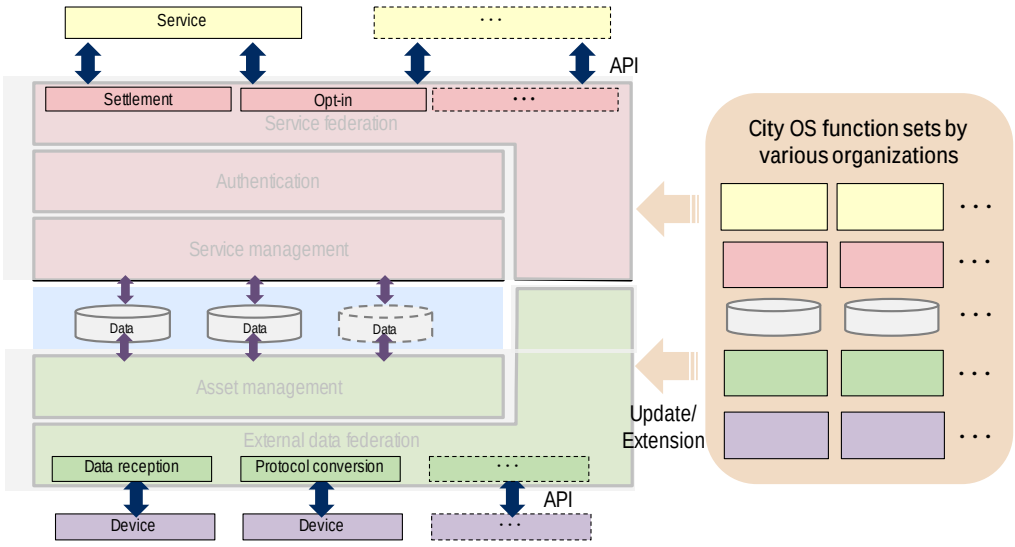


Figure 7.1-7 An example of construction by building block method

7.1.2 Determination of components of City OS architecture

With the understanding of characteristics of City OS and general concept of domestic and overseas Smart City, elements of City OS architecture are derived.

7.1.2.1 General concept of Smart City in Japan

City OS architecture is developed in reference to the definitions given in Society 5.0 proposed in the 5th Science and Technology Basic Plan. On the Science and Technology Policy page of Cabinet Office web site, it is described as follows:

"The society enabled by Society 5.0 overcomes the issues and difficulties of sharing various knowledge and information by way of connecting all the people and objects via IoT (Internet of Things) and creating new values as never before."

"Society 5.0 is realized by the highly integrated system of cyber space (virtual space) and physical space (real space)."

Based on these statements, definitions of people and objects existing in cyber space and physical space of Smart City are first considered.

People in Smart City are the people themselves in the region where Smart City activities are taking place in physical space, and defined as users of Smart City in cyber space. Objects in Smart City are any and all objects in the region in physical space, and defined as devices (sensors, mobile devices, etc.) which can act as the sources and recipients of data in cyber space. In addition, services are defined as the mechanisms for the users to make use of the data transmitted to and received from the devices in cyber space of Smart City. These correspond to service providers and other systems in physical space.

Relationship between physical/cyber spaces and people/objects in Smart City

	Physical space	Cyber space
People	People themselves in the region	User
Objects	All the objects in the region	Device
	Servicers/Other system	Service

Figure 7.1-8 Relationship between physical/cyber spaces and people/objects in Smart City

Also, as an activity record of the people in cyber space of Smart City, service usage log is defined as the information relating to what service is utilized with what device by the user along with chronological information and location information. City OS is defined as the framework to manage user/device/service/service usage log in cyber space.

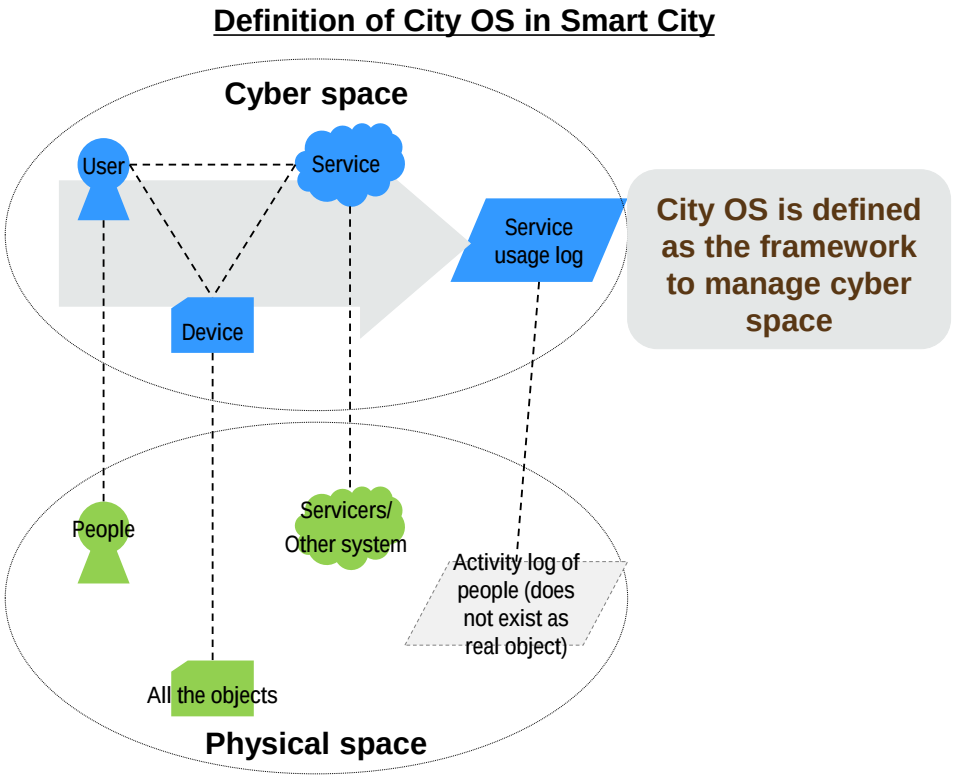


Figure 7.1-9 Definition of City OS

As a next step, City OS, which is the framework to manage user/device/service/service usage log is decomposed into the following seven architecture components.

- Authentication is defined as the architecture element which manages users by managing, authenticating, and authorizing user ID.
- Asset management is defined as management of devices which are treated as assets in Society 5.0 concept.
- External data federation is defined for data transmission to and from assets, which has to cope with the application of various data transmission technologies both old and new and the requirement for cross-sectoral data exchange.
- Service management is defined as the element which records the service usage log as well as manages the services themselves.

- Service federation is defined in order to guarantee API connection between services and City OS.
- Data management is defined for management of all types of data generated by five elements of authentication, asset management, external data federation, service management, and service federation, including service usage log.
- In addition to these six architecture components, operation function and security function which are necessary for ordinary system operations are defined.

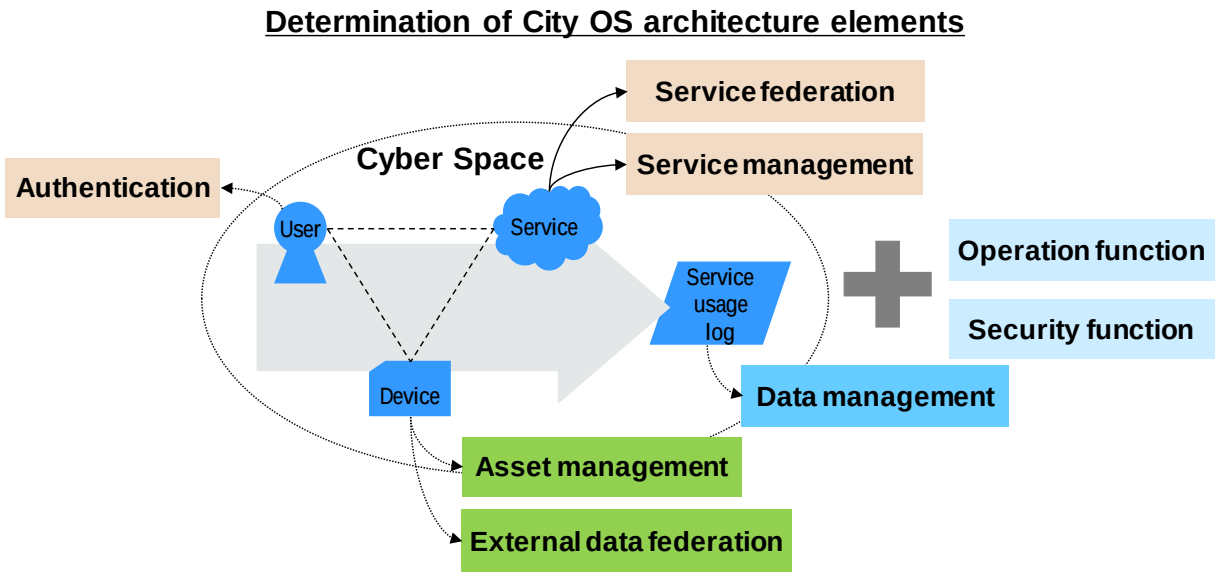


Figure 7.1-10 Derivation of City OS architecture components

7.1.2.2 Relationship between the characteristics of City OS and Smart City Reference Architecture

Figure 7.1-11 shows the relationship between the characteristics of City OS and Smart City Reference Architecture. The three characteristics of City OS are reflected as follows:

- Characteristic 1) The points reflected on the architecture from interoperability (connect) are authentication, service federation & service management functions, and Smart City asset (external data federation) & asset management functions, which constitute the basis for Interoperability,.
- Characteristic 2) The points reflected on the architecture from data exchange (flow) are data management and external data federation functions which enable brokering of external data.
- Characteristic 3) The points reflected on the architecture from scalability (future-proof) are the six components mentioned above, together with security and operation functions in added. The positioning of each element can be put in order by reflecting the characteristics of City OS onto the corresponding elements of the architecture.

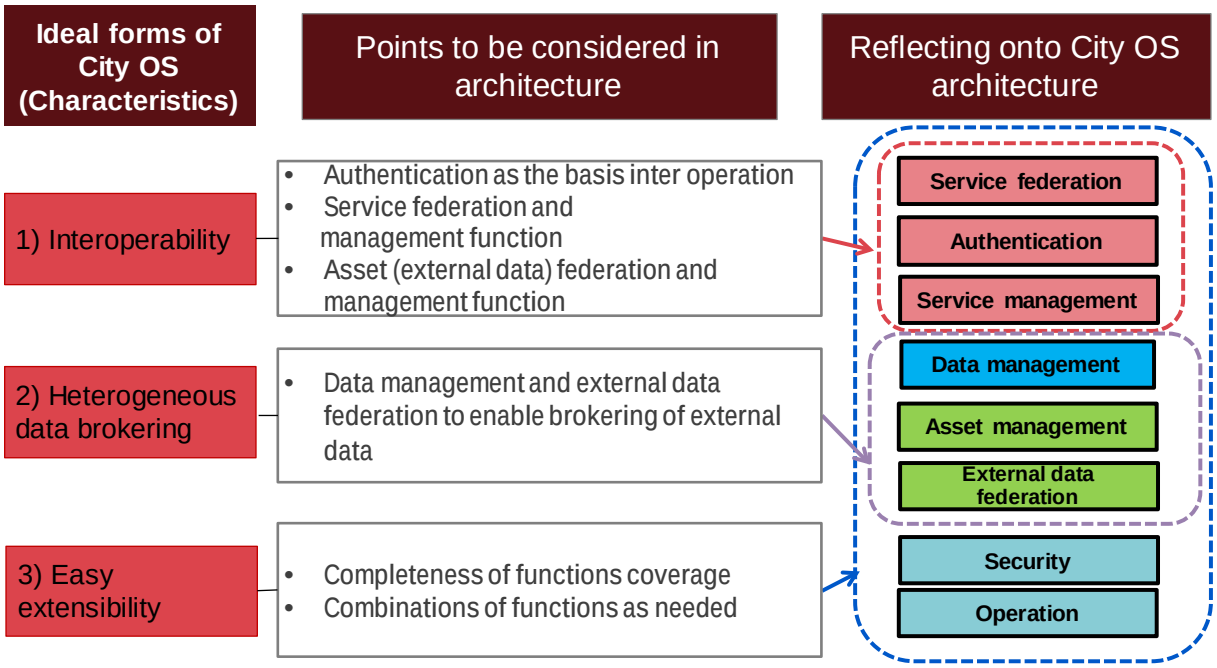


Figure 7.1-11 Relationship between the characteristics of City OS and Smart City Reference Architecture

7.1.2.3 Reference points of overseas Smart City architecture

City OS was designed with references to major overseas Smart City architectures. Table 7.1-4 shows the overview and reference points of each architecture. For more details of each architecture, please refer to “Appendix C Overseas Smart City Architecture”.

Table 7.1-4 Reference points of overseas Smart City architecture

Architecture	Overview	Reference points	Related chapter
SynchroniCity[35]	European IoT pilot project for Smart City, large scale initiatives with 20 cities currently participating.	- Components for the set functions of City OS and definitions thereof - Concept of API and data model in Minimal Interoperability Mechanisms, MIMs - Authentication-related API, data management-related API - Functions of architecture maintenance organization	7.2 Description of City OS functions 7.3 External federation 9.1.1 Various initiatives enabling maintenance and advancement of the architecture
FIWARE[36]	Platform software developed by FI-PPP for the purposes of enhancing competitiveness of Europe in the next generation internet technology and assisting development of smart applications in the social and public domains.	- Components of the functional groups of City OS and definitions thereof - Authentication-related API, data management-related API	7.2 Description of City OS functions 7.3 External federation
X-Road[37]	Platform for secure data exchange developed by Estonian Government.	- Functions of architecture maintenance organization	9.1.1 Various initiatives enabling maintenance and advancement of the architecture
IndiaStack[38]	Collective designation of Aadhaar and a set of APIs (e-KYC, e-Sign, etc.) to utilize Aadhaar, which was developed by Indian Government as the unified individual identification number utilizing biometric authentication technology.	- Authentication of individuals (Individual Authentication)	7.2.2 Authentication
IES-City[39]	Consensus framework established with the leadership of NIST (National Institute of Standards and Technology).	- Concept of Pivotal Points of Interoperability (PPI) which distills the critical technical points for ensuring interoperability	7.3 External federation

[35] https://synchronicity-iot.eu/

[36] https://www.fiware.org/

[37] https://x-road.global/

[38] https://www.indiastack.org/

[39] https://pages.nist.gov/smartcitiesarchitecture/

7.1.2.4 Smart City Reference Architecture of City OS

With the understanding of domestic and overseas Smart City concepts and characteristics of City OS as described above, City OS components of the determined Smart City Reference Architecture are described below. City OS consists of eight components which manage the storage of data collected from assets (various devices, systems, etc. utilized in the region), other systems, and other City OSs, and federations to services within and outside City OS. Figure 7.1-12 shows the entity relationship diagram of City OS.

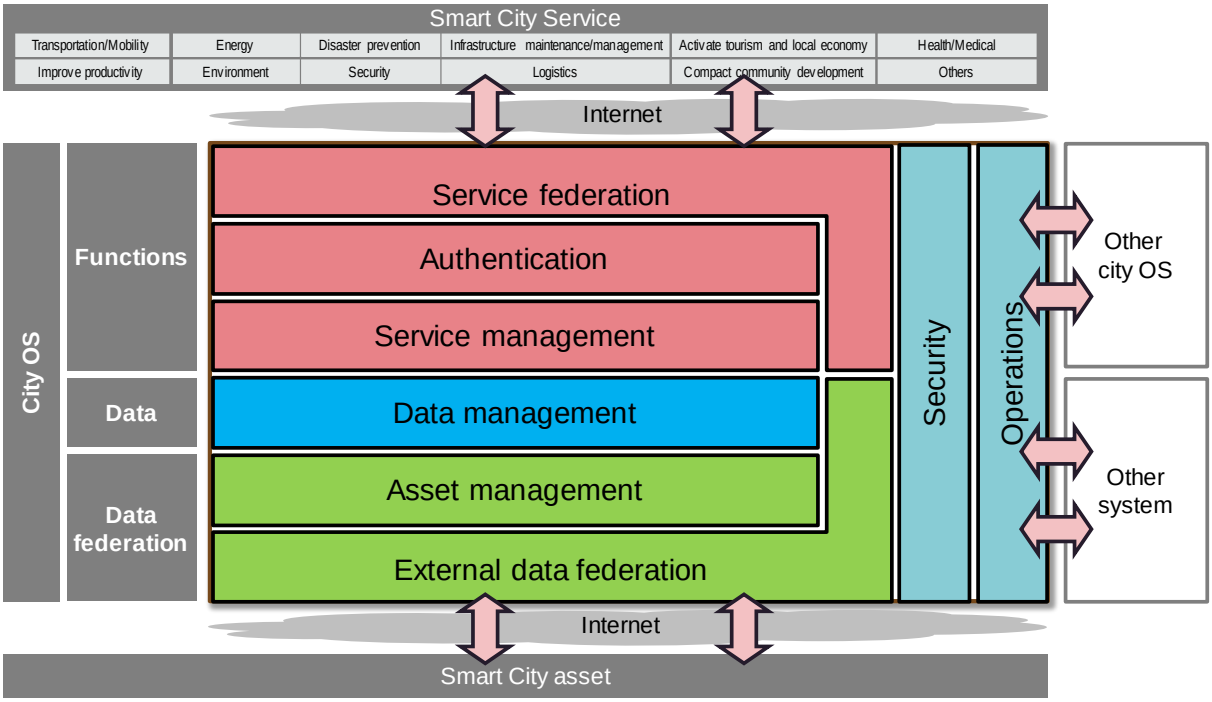


Figure 7.1-12 Entity Relationship Diagram of City OS

Table 7.1-5 shows the definitions of the eight function sets inside the red frame in Figure 7.1-12.

Table 7.1-5 Function sets overview of City OS

Structure Layer	Function Set	Definitions	Reference Chapter
Function (Service)	Service federation	Provide functions and API to federate with various services operating on City OS. Provide common services and open API, and have functions of API management and cross domain OS federation.	7.2.1
	Authentication	Provide appropriate authentication method for users, services and other City OS. Have functions of authentication & approval and user management.	7.2.2
	Service Management	Provide functions to manage Smart City services on City OS. Have functions of service management and service usage log management.	7.2.3
Data	Data Management	Provide functions to manage data stored and accumulated within City OS, and broker data distributed within and outside the region. Have functions of data brokering and data management.	7.2.4
Data Federation	Asset Management	Provide functions to manage Smart City assets and other systems federated to City OS, and execute Smart City asset control. Have functions for device management and system management.	7.2.5
	External Data federation	Provide functions to manage interface with Smart City assets or other systems, and absorb any differences in data formats and protocols. Have functions of data processing and data transmission.	7.2.6
Common Function	Security	Provide functions necessary to protect City OS against the internal and external threats.	7.2.7
	Operations	Provide system management function and management process necessary for the operations of IT systems on City OS.	7.2.8

In particular, with respect to the concept of interface in the conotext of federating City OS, please refer to “7.3 External federation”.

7.2 Descriptions of City OS functions

As shown in Figure 7.2-1, City OS consists of six function groups, namely “service federation”, “authentication”, “service management”, “data management”, “asset management”, and “external data federation”, and two additional types of function groups, “security” and “operation”, which are common to each function group, in total of eight function group.

Each function group contains function blocks grouping individual functions. These function blocks and the requirements for individual functions are described in the following sections. It should be noted that it is necessary that City OS is implemented with individual functions as a result of the selections of necessary requirements based on the issues to be resolved in the region and its goals in the future. These requirements are listed again in “Appendix A Summary of requirements of City OS” along with the examples of requirements selected based on the classification examples of City OS use cases.

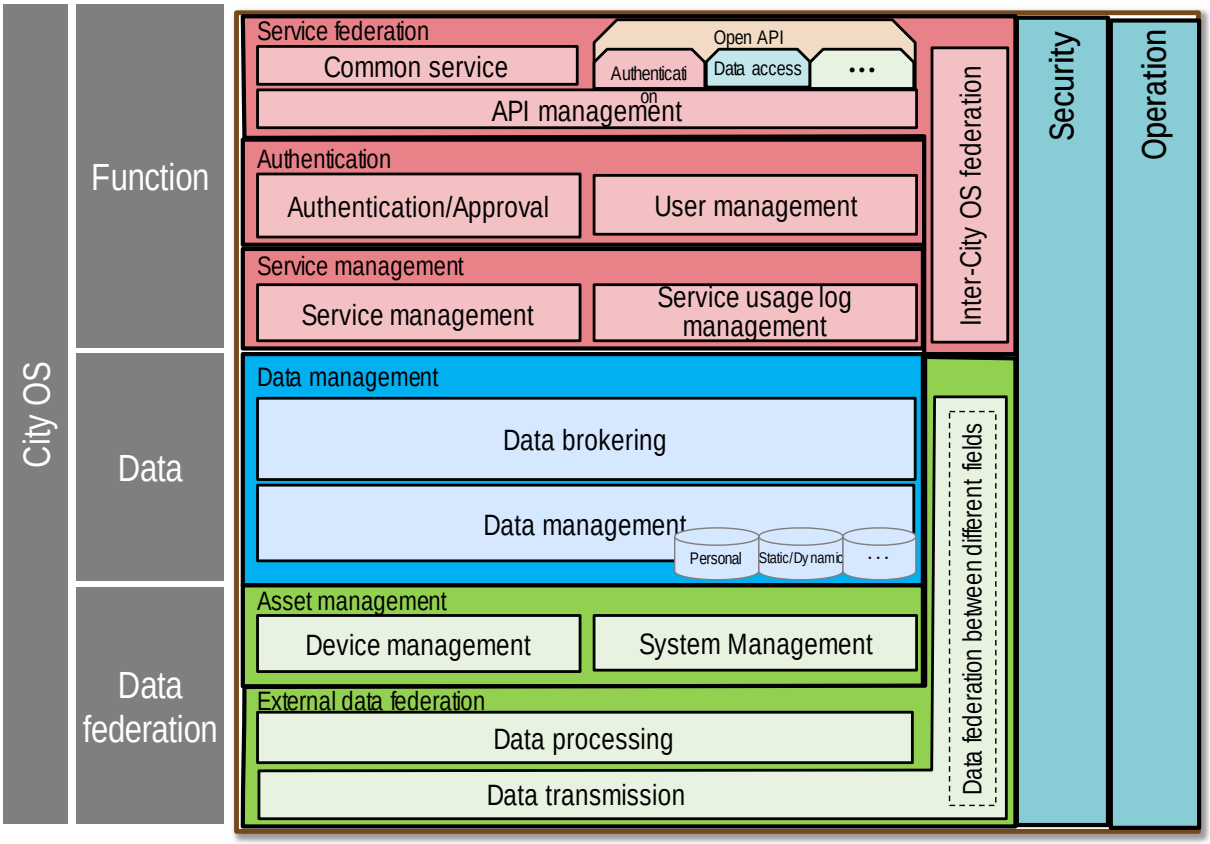


Figure 7.2-1 Details of City OS in Smart City Reference Architecture

7.2.1 Service federation

Service federation provides functions to federate various Smart City services, which operate on City OS, to City OS and other Smart City services. Service federation consists of “Common service”, “Open API”, “API management, and “Cross domain OS federation” functional blocks.

7.2.1.1 Description of functional blocks for service federation

1) Common service

Common service is a function of City OS to provide services which are used commonly across regions and domains as the cooperative area. It is often provided via user interface, and includes, for example, developer portal site for the users of City OS, bidirectional communication portal site to enable connection between residents and municipalities, etc.

2) Open API

Open API is the interface specifications of APIs provided by City OS that are made public for external use by other organizations. It is made open to public and enables external access by working with API management, which includes authentication, and data access, etc. required for service federation and cross domain OS federation. Open API is sometimes utilized to federate with Smart City assets and other systems within external data federation.

3) API management

API management manages and publishes various functions provided by City OS as API. API management covers not only the API made available for external access but also the API exclusively used within City OS. It includes API life cycle management for managing and publishing API, API gateway to control the flow of API, etc.

4) Cross domain OS federation

Cross domain OS federation is a function to federate various functions, data, etc. belonging to City OS with another City OS. It includes authentication federation which enables the use of the same credentials information when a user moves to another city, and data federation to share data between City OSs.

7.2.1.2 Functional requirement

Table 7.2-1 shows functional requirement for common services.

Table 7.2-1 Functional requirement for common services

No.	Individual function	Description
1	Developer portal site	For City OS users, catalog functions to enable API and data searches & publishing specifications, and console functions to enable evaluation of API, etc. must be provided.
2	Bidirectional communication portal site	For residents and municipalities, a function must be provided to enable aggregation, delivery, etc. of region-related services and information. It should be equipped with a function to enable bidirectional communication by connecting residents and municipalities, and residents and Smart City services, utilized for the resolution of issues and the improvement in convenience and quality.
3	Personalize	In order to present Smart City services meeting the preferences of residents, a function must be provided to prioritize the display order of articles reflecting the interest of individual residents.
4	Contents management	A function must be provided to enable creation, delivery, etc. of contents to be published on the portal sites and home pages provided by municipalities. It should be equipped with a campaign management function to enable the measurement of the effects of staged events, mail deliveries, etc.
5	Regional points management	A function must be provided to enable and manage a point program service unique to each region aiming at leading and maintaining residents' participation in the regional issues.
6	Opt-in management	A function must be provided for residents to decide by themselves to designate the permitted extent of the publication of individual personal data to City OS operators and service providers.
7	Visualization/analysis dashboard	A dashboard function must be provided to enable visualization & analysis of the status of the city utilizing the data within and outside City OS for the purpose of resolving regional issues by residents and municipalities. It should be possible to measure the effects of the measures, such as the analysis based on the KGI/KPI set in the strategy.

As for functional requirement for open API, please refer to “7.3 External federation”. In particular, “7.3.2.1 Authentication-related API” and “7.3.2.2 Data management-related API” are necessary for cross domain OS federation.

Table 7.2-2 shows functional requirement for API management.

Table 7.2-2 Functional requirement for API management

No.	Individual function	Description
1	API lifecycle management	Manage the lifecycle (register, refer, update, delete) of City OS API.
2	API gateway	Execute API usage restriction, network throughput restriction, aggregation of multiple APIs, etc.

Table 7.2-3 shows functional requirement for cross domain OS federation.

Table 7.2-3 Functional requirement for cross domain OS federation

No.	Individual function	Description
1	Authentication federation	Respond to the authentication request from the user based on the user authentication information on the other City OS with federation to the other City OS. It should be provided as a function to enable cross domain authentication federation in accordance with the functional requirement as described in “7.3.2.1 Authentication-related API”.
2	Data brokering	Provide the data of other City OS to users by federating with other City OS. It should be noted that this function is equivalent to what is defined in “7.2.4.1 (1) Data brokering”. Data exchange between City OSs is enabled by publishing and federating API in accordance with the functional requirements described in “7.3.2.2 Data management-related API”.

7.2.2 Authentication

Authentication is a set of functions for City OS to provide appropriate authentication methods for users of City OS, applications and other systems which are federated with City OS. Authentication consists of “authentication & approval”, and “user management” functional blocks.

7.2.2.1 Description of function blocks for authentication

(1) Authentication & approval

Authentication & approval can be broadly categorized into authentication function and approval function.

Authentication function is a function to identify who the user of City OS is. Based on the identification of the user by this function and the confirmation of the usage authority by the approval function, it is possible to determine which functions of City OS can be used.

The users of City OS are varied and include applications, etc. as well as users. It is necessary to be able to provide an appropriate authentication method for each of these cases. In addition, a variety of entities use this authentication function to implement Smart City services on City OS. This authentication function is required to provide a secure and easy-to-use interface to external Smart City services.

Approval function is a function which enables user-by-user or role-by-role setup and determination of usage authority and usage period of the data and services managed by City OS. It hereby enables prevention of improper data access and service use. For approval of data access, it enables approval of data disclosure range and access duration. For approval of service access, it enables approval of usage restrictions and expiration date of the service.

(2) User management

User management is a function to centrally manage the users of City OS. It hereby enables the users of City OS to access various applications implemented on City OS by the same single user information. User management enables management of authentication information (password, credentials, etc.), attributes information (name, affiliation, etc.), in association with ID identifying the user, and of ID lifecycle.

7.2.2.2 Functional requirement for authentication

Table 7.2-4 shows functional requirement for authentication & approval.

Table 7.2-4 Functional requirement for authentication & approval.

No.	Individual function	Description
1	Authentication	Certify legitimacy of the user from eligibility information (user ID, password, biometric information, etc.) stored under “user management”, and identify the account.
2	Approval	In association with “user management”, permit or restrict the usage of various functions of City OS and managed data based on the roles and policies associated with the account.
3	Individual authentication	In the case of using personal data, securely confirm the identity by multi-factor authentication (a combination of biometric authentication and individual number card, etc.) * There may be cases in which individual authentication is not implemented on City OS but managed by each respective service.
4	Single sign-on	Enable single sign-on by centralized management of authentication for multiple services federated to City OS. Once authenticated, it should be possible for users not having to be authenticated for each one of the Smart City services federated to City OS, hence one-stop service is realized.

Table 7.2-5 shows functional requirement for user management.

Table 7.2-5 Functional requirement for user management

No.	Individual function	Description
1	Account management	Manage authentication information (password) and attributes information (name, organization, etc.) in association with a specific ID identifying the user, and manage ID lifecycle (register, refer, change, delete).
2	Role management	Manage a role that defines the group the user belongs to (user, administrator, etc.).
3	Policy management	Manage the control policies which define the extent and authority to access City OS, separately from accounts and roles.

7.2.3 Service management

Service management is a set of functions which manage and appropriately operate Smart City services federated with City OS. Service management consists of “service management” and “service usage log management” functional blocks.

7.2.3.1 Description of functional blocks for service management

(1) Service management

Service management is a function to manage and publish services which are federated with City OS. Service providers can be either City OS operators or service providers.

It also manages the relationship between users and Smart City services, and further enables the users to change the mode of service usage based on subscription approach.

(2) Service usage log management

Service usage log management is a function to store users’ usage status of the Smart City services federated with City OS. Stored service usage log can be utilized to provide optimum personalized services in accordance with the interest of individual user assuming opt-in by the user.

7.2.3.2 Functional requirement for service management

Table 7.2-6 shows functional requirement for service management.

Table 7.2-6 Functional requirement for service management

No.	Individual function	Description
1	Service lifecycle management	Manage the lifecycle (register, refer, update, delete) of Smart City services federated with City OS. A list of services managed by City OS should be published to users along with “Service federation”.
2	Subscription management	Manage the status of subscription (start & end of usage, change setup for usage authorization) for the Smart City services available for use by users.

Table 7.2-7 shows functional requirement for service usage log management.

Table 7.2-7 Functional requirement for service usage log management

No.	Individual function	Description
1	Usage log management	Provide a function to store and publish user's City OS and Smart City service usage logs upon user's approval.

7.2.4 Data management

Data management is a set of functions to enable management of data stored and accumulated on City OS, and brokering of data distributed across standalone cities, multi-cities, and other systems. Data management consists of “data brokering” and “data management” functional blocks.

7.2.4.1 Description of functional blocks for data management

(1) Data brokering

Data brokering is a function to broker data access by managing the location information of data existing within and outside City OS. It provides a function for users to transparently access the data stored on City OS and the data located in other City OSs and other systems using a single interface.

(2) Data management

Data management is a function to store and accumulate data collected through service federation and external data federation irrespective of data categories or data formats.

City OS is required to manage various types of data of different characteristics (variations, frequency of updates, volume) for each data category. When handling personal data, in particular, it facilitates tamper-free data storage and transaction management.

In order to interoperate data between different City OSs, it is necessary to manage data by global and unique ID. It hereby becomes possible to identify a particular piece of data out of various data across regions.

7.2.4.2 Functional requirement for data management

Table 7.2-8 shows functional requirement for data brokering.

Table 7.2-8 Functional requirement for data brokering

No.	Individual function	Description
1	Data storage	Process (register, refer, update, delete) data managed by City OS by federating with “data management”.
2	Data exchange	Broker (register, refer, update, delete) data distributed across other City OSs and other systems.
3	Event processing	Execute real-time processing of the data brokered by City OS in accordance with the pre-defined policies. It hereby becomes possible to provide a mechanism to switch functions dynamically and flexibly between analysis/conversion/processing of the data distributed over within and outside City OS and change in access authorization reflecting the changes in social environment.

Table 7.2-9 shows functional requirement for data management.

Table 7.2-9 Functional requirement for data management

No.	Individual function	Description
1	Data store	In dealing with various data of different characteristics (diversity, update frequency, volumy), appropriately store and utilize data necessary for the resolution of the issues in the region. The data classification includes personal data, real-time data, etc. It should be possible to manage logs to chronologically verify such sequential data as real-time data.
2	Unique ID management	Manage ID unique to each piece of data on City OS, and provide a mechanism to enable identification of a particular piece of data out of various data across regions. It is recommended to adopt regional domains, etc., as the unique ID has to be globally unique.

7.2.5 Asset management

Asset management is a set of functions which manage data collection, and registration, deletion, etc. of Smart City assets & other systems to be connected, and execute control over Smart City assets. Asset management consists of “device management” and “system management” functional blocks.

7.2.5.1 Description of functional blocks for asset management

(1) Device management

Device management is a function to manage and monitor the status of IoT devices, etc. connected to City OS, so that system administrator can detect connection anomaly and other problems of devices. This function enables centralized remote management and monitoring of devices installed across the real world. Also, remote management for control (actuation), maintenance, etc. of devices are enabled by way of sending control instructions for reboot, firmware update, etc.

(2) System management

System management is a function to manage information required for interfacing with other systems such as authentication information, connection information, contract agreement information, etc. of other systems connected to City OS, and also for systems administrators to manage data federation status or connection status with other systems. This function enables management of data collection, brokering and its status with various systems connected to City OS.

7.2.5.2 Functional requirement for asset management

(1) Device management

Table 7.2-10 shows functional requirement for device management.

Table 7.2-10 Functional requirement for device management

No.	Individual function	Description
1	Device lifecycle registration	Manage the lifecycle (register/refer/update/delete) of device information (device ID, unique MAC address, etc.)
2	Device status management	Manage and publish device status (operation status, equipment information, etc.) for the devices registered
3	Device control (actuation)	Transmit commands to control devices, such as reboot and change operations of the devices connected
4	Device monitoring	Monitor operational-or-non-operational status of the devices connected, or failure incidents transmitted by the system
5	Device authentication	Permit access only from those devices registered in advance

(2) System management

Table 7.2-11 shows functional requirement for system management.

Table 7.2-11 Functional requirement for system management

No.	Individual function	Description
1	System lifecycle registration	Manage the lifecycle (register/refer/update/delete) of the federation information of other systems federated to City OS. Also, it should be possible to manage authentication methods and their credentials information, as other systems often require authentication.
2	System status management	Manage and publish connection status (operation status, equipment information, etc.) with other systems, for other systems registered.

7.2.6 External data federation

External data federation is a set of functions which manage the interface between City OS and Smart City assets as well as other systems, and absorbs the mismatch in data models and protocols. External data federation consists of “data processing” and “data transmission” functional blocks. “Data transmission” also includes the concept of enabling data federation across different domains and regions via what is called “cross-sectoral data federation (cross-sectoral data federation connector)”.

7.2.6.1 Description of functional blocks for external data federation

(1) Data processing

Data processing is a function to absorb mismatch in data models collected from Smart City assets and other systems, and convert them into standard data models, etc. This function enables more extensive and generic utilization of the collected data, and realizes active usage of the data between different domains. For data processing function, there is no difference in requirements between Smart City assets and other systems.

(2) Data transmission

Data transmission is a function to absorb mismatch in interface protocols. It supports system connection under multiple interface protocols, and copes with various connection requests coming from Smart City assets and other systems. By absorbing mismatch of interface protocols with this function, data can be federated via standard interface in the upper layer. For data transmission function, requirements for Smart City assets are different from those for other systems.

In the case of interfacing with Smart City assets, bidirectional communication protocol is required in order to support transmission of control commands to devices. Also, due to limitations of hardware resources and communication environment, light-weight communication method is preferred.

In the case of interfacing with other systems, the target systems are often IT systems already in existence with a set of custom interfaces. In order to accommodate such custom interfaces, a mechanism to accept a broad range of interface protocols is required. Also, it is necessary to have mechanisms to ensure data interoperability with other City OS (transmission of collected data and data access requests, etc.).

(3) Cross-sectoral data federation (cross-sectoral data federation connector)

Currently, when City OS intends to send and receive data to and from various external systems, highly cumbersome and resorce-consumig labor is needed to check and adjust interface specifications erery time a new external system is to be federated. This issue will be resolved by implementing cross-sectoral data federation connector on both City OS and other systems to be federated. Such a universal framework will easily provide interoperability and achieve smooth data federation.

Cross-sectoral data federation connector is currently under research and development projects,[40] and this connector is expected to provide functions for distributed data search, data exchange control, and data exchange log, etc. in the future. In particular with respect to data exchange log, it is not simply recoding the data exchange logs but expected to ensure the authenticity in terms of data quality by way of block-chain technology. From the standpoint of ease of development for external data federation functions, City OS should also be equipped with a function to interface with cross-sectoral data federation connector in the future.

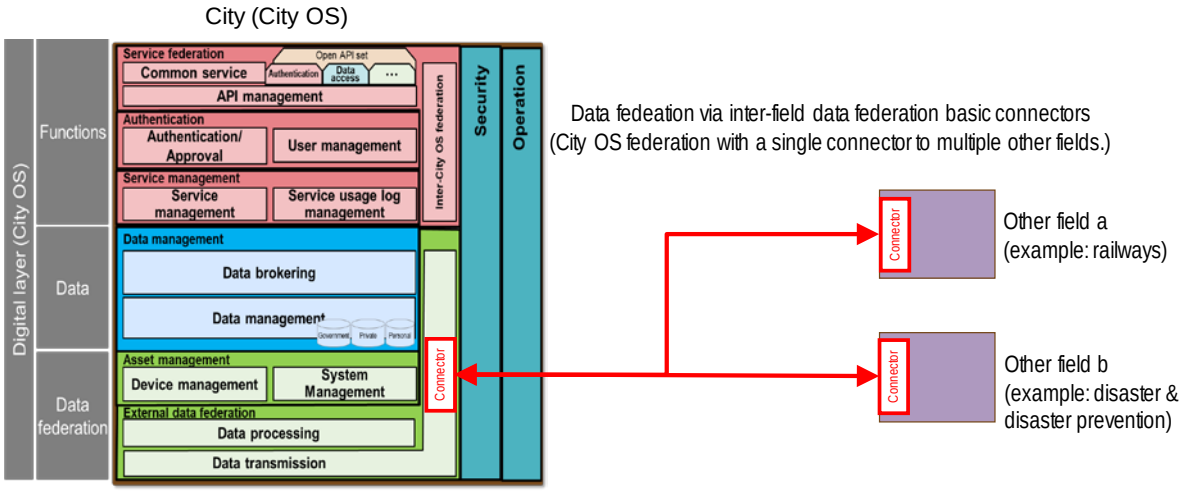


Figure 7.2-2 Data federation diagram with other fields via inter-field data federation basic connectors

7.2.6.2 Functional requirement for external data federation

Table 7.2-12 shows functional requirement for data processing.

[40] Cross-ministerial Strategic Innovation Promotion Program (SIP) Second Phase / Big-data and AI-enabled Cyberspace Technologies / Research on the infrastructure to realize data utilization and service provision overcoming the boundaries of fields and organizations

Table 7.2-12 Functional requirement for data processing

No.	Individual function	Description
1	Data conversion	Convert externally acquired data into the format that City OS can handle. Terms to be converted include vocabulary, format, subject matter, etc. and vary depending on the data to be handled.
2	Data acceptance (queuing)	Accept data access (register, refer) to accumulate data on City OS. Federated target includes Smart City assets, other systems, etc.
3	Data acquisition (crawling)	Regularly crawl other systems and acquire data.
4	Data complement	Complement missing data in real-time data, etc. and improve data quality. There are various data complement methods, it should be possible to prepare complement method options in order to meet every purpose.

Table 7.2-13 shows functional requirement for data transmission.

Table 7.2-13 Functional requirement for data transmission

No.	Individual function	Description
1	Protocol conversion	In order to interface with Smart City assets and other systems operated across the region, convert the standard communication protocols into the communication protocol that City OS operates with.
2	Cross-sectoral Data search	Search the data distributed over external City OS based on the summarized information of data (data catalog). * To be utilized for cross-sectoral data federation connectors in the future.
3	Cross-sectoral Data exchange control	Control extent of data access by checking data usage authorization based on mutual agreement rules set between City OS and other systems. * To be utilized for cross-sectoral data federation connectors in the future.
4	Cross-sectoral Data exchange record	Record the data exchange logs mutually federated between City OS and other systems, in order to improve data quality through traceability. * To be utilized for federations to cross-sectoral data federation connectors in the future.

7.2.7 Security

Security is a set of functions to facilitate protection of City OS against threats from outside City OS and vulnerabilities within City OS.

7.2.7.1 Description of functional blocks for security

Security measures of City OS are divided into two main categories of 1) technical measures, and 2) managemental measures (including personnel security, organizational security, physical (environmental) security).

Technical measures include authentication, encryption, unauthorized access prevention, unauthorized access detection and interception technology, etc., and are defined as required functions to be implemented in City OS.

Managemental measures include such personnel security as education and making rules, such organizational security as vulnerability management and security audit, and such physical security as facility access management and vandalism prevention, and are defined as necessary items for operation and management of City OS.

7.2.7.2 Functional requirement for security

As technical measures, City OS incorporates individual functions as shown in Table 7.2-14.

Table 7.2-14 Functional requirement of technical measures

No.	Individual function	Description
1	Authentication	Provide a function to authorize digital access by way of verifying whether the access requester is legitimate or not, for users, Smart City services, other City OS, other systems, IoT devices, etc. which interfaces with City OS. This function is the same as what is defined in the following chapters: - “7.2.2 Authentication”. Authentication for users, Smart City services, and other City OSes - “7.2.5 Asset management” Device authentication for Smart City assets
2	Encryption	Apply security encryption appropriate for each confidentiality level, to the communication by City OS (communication within City OS and communication with the outside world of City OS) and the data managed by City OS.

No.	Individual function	Description
3	Unauthorized access prevention	Provide a function to block unpermitted communication (packets with unauthorized IP address and port numbers, etc.) for all the communication by City OS. It is also called a firewall function.
4	Unauthorized access detection/interception function	Provide a function to detect and intercept DoS attacks and those attacks targeting the vulnerability of application layer, etc. which cannot be dealt with the unauthorized access prevention function.

Figure 7.2-3 shows the relationship between individual security function and various functional blocks of City OS.

Security function / Functional block		Authentication	Encryption	Unauthorized access prevention	Unauthorized access detection/ interception
Functions	Service federation	–	○ (encrypt communication)	○	○
	Authentication	○ (defined in authentication)	–	–	–
	Service management	–	–	–	–
Data	Data management	–	○ (encrypt data)	–	–
Data federation	Asset management	○ (authentication for assets)	–	–	–
	External data federation	–	○ (encrypt communication)	○	○

Figure 7.2-3 Relationship between individual security function and each functional block of City OS

As managemental measures, management and operation of City OS is required to satisfy the requirements shown in Table 7.2-15.

Table 7.2-15 Functional requirements of managemental measures

No.	Individual function	Description
1	Vulnerability management	For software which makes up City OS, information relating to its vulnerability is to be collected and necessary measures such as application of patches as needed are executed. Also, vulnerability test is to be conducted for City OS on a regular basis and countermeasures in accordance with the result are executed.
2	Log management	Acquire logs of communication and processes carried out by City OS. The acquired logs are to be stored for a specified duration in order to preserve evidences.

As City OS is implemented as a platform on the cloud, it is also desirable to refer to "Information Security Program Guideline for Provision of Cloud Services (Version 2)"[41] published by Ministry of Internal Affairs and Communication as general information to be considered, and execute both technical measures and managemental measures.

[41] Information Security Program Guideline for Provision of Cloud Services (Version 2)
https://www.soumu.go.jp/main_content/000566969.pdf

7.2.8 Operation

Operation is a set of functions to provide system management and management process required for maintenance and advancement of City OS. The system management and management process are indispensable for the continuous advancement of City OS itself after the initial implementation of City OS. Thanks to them, by further improving the common services and various functions, City OS is able to accommodate various Smart City services leading to continuous maintenance and advancement of City OS.

7.2.8.1 Description of functional blocks for operation

(1) System management

It is to cope with the non-functional requirement to maintain and manage City OS in order for City OS to stably provide various functions without system outage due to unexpected causes. It includes ensuring availability for prompt detection and recovery of failures, ensuring system scalability by building loosely coupled systems, etc.

(2) Management process

It specifies processes and rules required for the continuous efforts towards further advancement of City OS. It includes service transition management in preparation for going live with Smart City services and City OS, system operation management to ensure the quality of City OS, etc.

7.2.8.2 Functional and non-functional requirements for operation

Table 7.2-16 shows functional requirements for system management.

Table 7.2-16 Functional requirements for system management

No.	Individual function	Description
1	System scalability	Provide a mechanism to allow for continual and easy additions and revision of functions for future in response to the issues to be resolved in the region and the goals to aim at in the future. It should be possible to flexibly accommodate the reconfiguration of functions by building loosely coupled systems via the likes of building block method.
2	Availability	Provide a mechanism for City OS to continuously operate robustly without outage in the event of a failure of City OS. It is important to minimize the impact to users by defining the service level of City OS, prompt detection and recovery of failures, implementing redundancy, etc.

Table 7.2-17 shows requirements for management process.

Table 7.2-17 Requirements for management process

No.	Individual process	Description
1	City OS planning and development management	Plan and develop extension of various functions of City OS in accordance with the service expansion due to the growth of the region, etc. Based on the plan, the implementation plan for new common services and new functions is formulated, and the process of defining the requirements, design, development, test, and transition is managed. It is desirable to adopt not only the traditional waterfall-style development but also the agile-style development process which enables rapid implementation of various functions.
2	Service transition management	Formulate and manage the plan to prepare and transition Smart City services and various functions, in the event of going live with various functions of Smart City services and City OS.
3	System operation management	Define management tools and processes for the operation (change management, configuration management, incident management, operation service management, capacity management, etc.) of City OS.

In order to construct the optimal system configuration in response to the requested service level, "Non-functional Requirement Grade"[42] published by IPA/ISEC (Information-technology Promotion Agency/IT Security Evaluation and Certification) should be used. It is desirable to define the non-functional requirements of City OS in accordance with the importance of Smart City services on each City OS, and configure the system meeting these requirements.

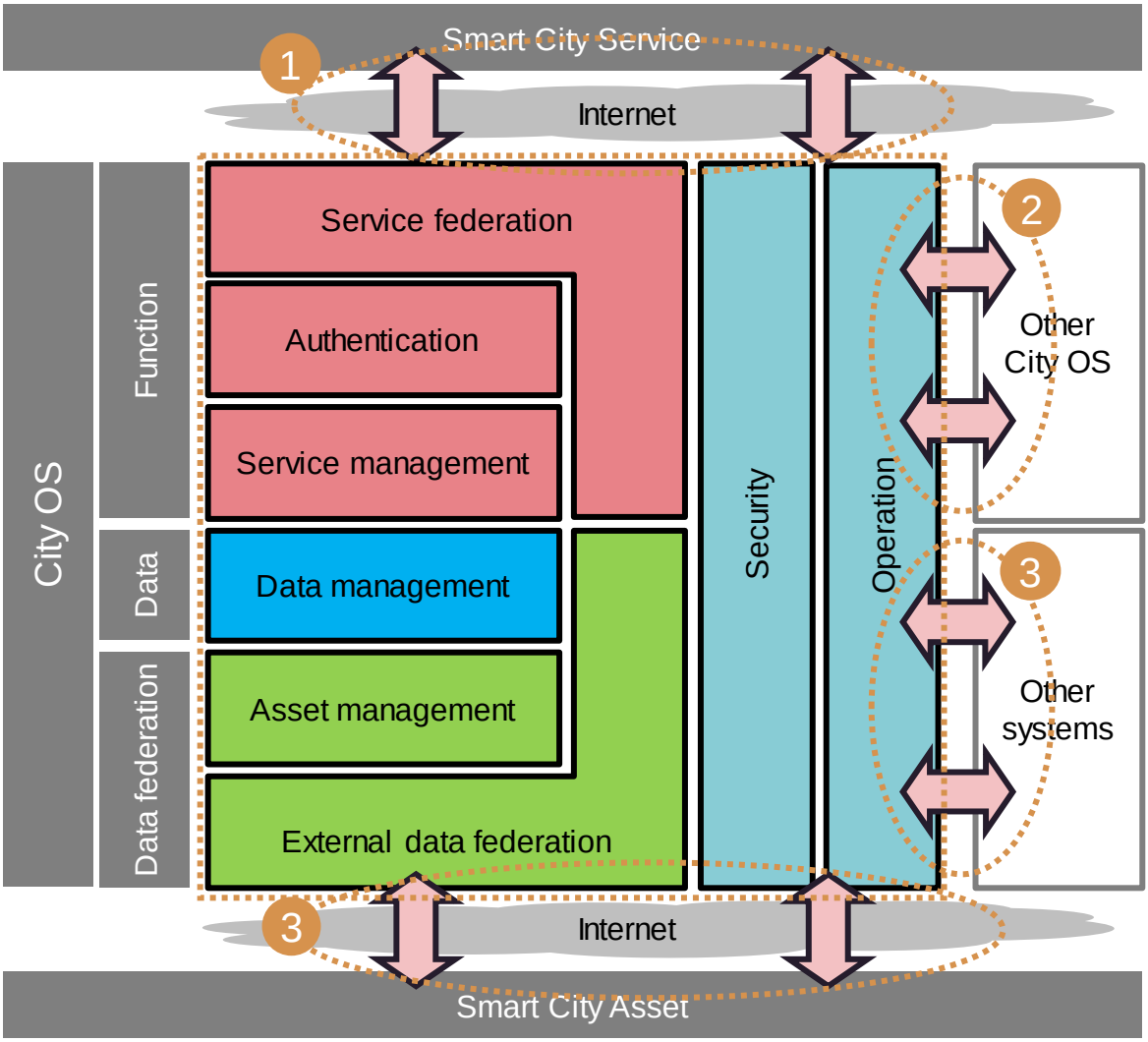
[42] Non-functional Requirement Grade https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html

7.3 External Federation

7.3.1 External federation of City OS and the concept of API

7.3.1.1 External federation methods

Figure 7.3-1 shows types of external federation supported by City OS, namely 1) service federation, 2) cross domain OS federation, and 3) asset /other system federation. As for federation methods, there are API-based federation and functional federation. As for Smart City assets and other systems, please refer to “8. Smart City asset and other systems”.



No.	Federation method	Description	Federation point
1	Service federation	Publish interfaces and common services which provide various functions to Smart City services operating on City OS.	Define data/communication method/authentication and API required for inter-service federation.
2	Cross domain OS federation	Federation with Smart City services and data published by another City OS.	Define data/communication method/authentication and API required for cross domain OS federation.
3	Asset federation/ Other system federation	Collect and broker data for various Smart City assets and other systems.	Define data/communication methods and interfaces required for federation with Smart City assets and other systems.

Figure 7.3-1 Federation points for City OS

In order to enable service federation, cross domain OS federation, and federation with assets & other systems for City OS, interoperability of API and data is required. Interoperability of City OS is described here with reference to practices in other countries.

7.3.1.2 Oversea's initiatives for interoperability

As examples found in oversea countries, the framework for interoperability of public authorities in Europe (EIF) and the concept of sharing only the technically critical parts out of the overall architecture (MIMs, PPI) are described.

(1) European Interoperability Framework (EIF)

The concept of interoperability for public agencies in Europe is proposed as European Interoperability Framework (EIF)[43]. Even though it is intended for administrative entities, its concept provides a good reference for Smart City.

Of particular importance for City OS is "Semantic Interoperability", specifically the data models (data, structure, items, etc.) related to data exchange, and "Technical

[43] European Interoperability Framework (EIF)
https://ec.europa.eu/isa2/sites/isa/files/eif_brochure_final.pdf

Interoperability", specifically the communication protocols technically necessary for interfaces.

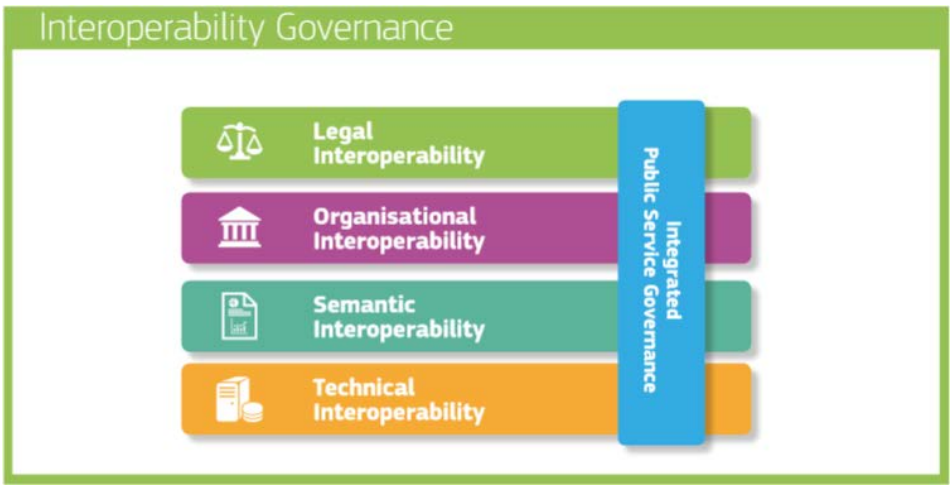


Figure 7.3-2 European Interoperability Framework (EIF)

(2) Minimal Interoperability Mechanisms (MIMs)

Open & Agile Smart City (OASC), which is an international organization for Smart Cities originated in Europe, is proposing the concept of minimal interoperability mechanisms (Minimal Interoperability Mechanisms, MIMs)[44].

During Connected Smart City Conference 2020 (1/23, Paris), an annual assembly conducted by Open & Agile Smart City (OASC), Personal Data Management proposed by Helsinki and Fair AI proposed by Amsterdam were newly added to MIMs in addition to the three existing MIMs that were approved and established earlier as shown in Table 7.3-1.

[44] https://www.youtube.com/watch?v=Dkq8X0K-iwY

Table 7.3-1 Minimal Interoperability Mechanisms, MIMs

MIM	Point	Description	References	Related Standards &[Baselines]
OASC Context Information Management MIM	Context Info Management API	This API allow to access to real-time context information from the different cities.	Reference Architecture for IoT-Enabled Smart Cities[SC-D2.10]	ETSI NGSI-LD prelim API, OMA NGSI, ITU-T SG20'/FG-DPM'
OASC Data Models MIM	Shared Data Models	Guidelines and catalogue of common data models in different verticals to enable interoperability for applications and systems among different cities.	Guidelines for the definition of OASC Shared Data Models[SC-D2.2] Catalogue of SASC Shared Data Models for Smart City domains [SC-D2.3]	[FIWARE, GSMA, Schema.org, SAREF, Synchronicity RZ+ partner data models]
OASC Ecosystem Transactions Management MIM	Marketplace API	It exposes functionalities such as catalogue management, ordering management, revenue management, Service Level Agreements (SLA), License management etc. Complemented by marketplaces for hardware and services.	Basic Data Marketplace Enablers(SC-D2.4)	TM Forum API

In particular, SynchroniCity, which is formulated based on the concept of minimal interoperability mechanisms, has set the requirements for primary interfaces known as interoperability points in the architecture. The interoperability points are the main interfaces to federate applications and systems of each city to Synchronicity framework, and consist of APIs relating to provision and use of data between domains/cities, and specifications and guidelines for common data model.

Table 7.3-2 shows APIs of interoperability points in SynchroniCity[45].

[45] https://synchronicityiot.docs.apiary.io/

Table 7.3-2 APIs in SynchroniCity

No.	API	Description
1	IoT Management	Absorb differences in multiple standards and protocols of various IoT devices, and make them compatible and available to the SynchroniCity framework.
2	Context Data Management	As the core of the architecture, manage context information from various IoT devices and data sources, and provide unified approaches and interfaces.
3	Data Storage Management	Provide unified access and data management for wide variety of data stored in various data storages, and functions related to data quality assurance (data cleansing, data quality checking tools, etc.)
4	IoT Data Marketplace	Implement a system (data trade marketplace) to exchange various data, and provide such functions as asset cataloging, order placement, profit/customer/SLA/license management, etc.
5	Security, Privacy and Governance	Provide essential security functions such as ID management, confidentiality, authentication, approval, completeness, disapproval prevention, access control, etc. by covering all aspects of security related to data and platform services in the architecture.

Figure 7.3-3 shows an example of common data model[46].

Vertical	Data Model	Description	Original Source	Approval Status
Environment	AirQualityObserved	It represents an observation of air quality conditions at a certain place and time	GSMA	Approved
Environment	NoiseLevelObserved	It represents an observation of those parameters that estimate noise pressure levels at a certain place and time	FIWARE	Approved
PointOfInterest	PointOfInterest	A harmonised geographic description of a Point of Interest	GSMA updated by SynchroniCity	Approved

Define details of covered data models and descriptions

※一部を記載

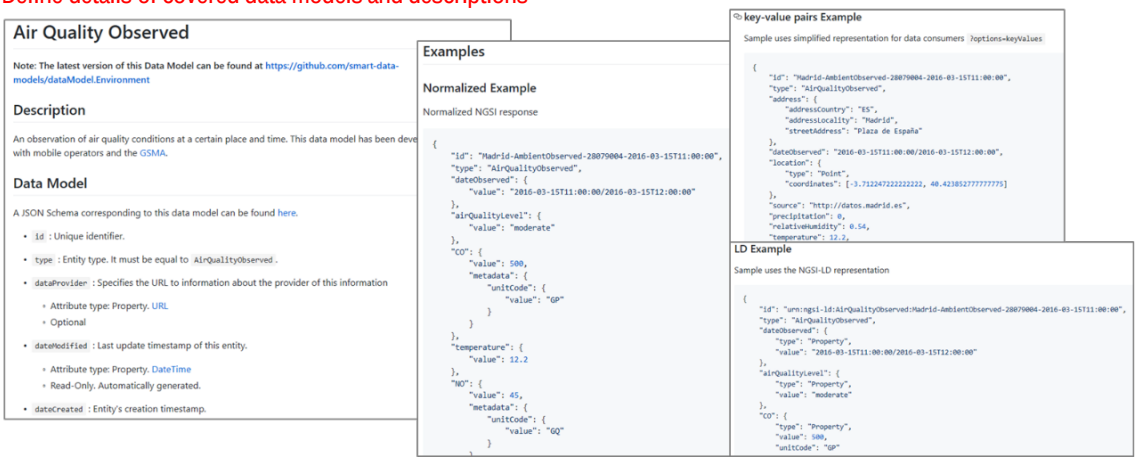


Figure 7.3-3 Example of data model standardization in SynchroniCity

(3) Pivotal Points of Interoperability (PPI) [47]

The National Institute of Standards and Technology (NIST) in the United States has called out to various IoT platform stakeholders in order to realize effective and powerful Smart City solutions, and conducted an international collaborative effort on the comparative studies of architecture (IES-City Framework) to develop a framework for building consensus on the characteristics of the architectural as shown in Figure 7.3-4.

[46] https://gitlab.com/synchronicity-iot/synchronicity-data-models

[47] NIST A Consensus Framework for Smart City Architectures IES-City Framework https://s3.amazonaws.com/nist-sgcps/smartcityframework/files/ies-city_framework/IES-CityFramework_Version_1_0_20180930.pdf

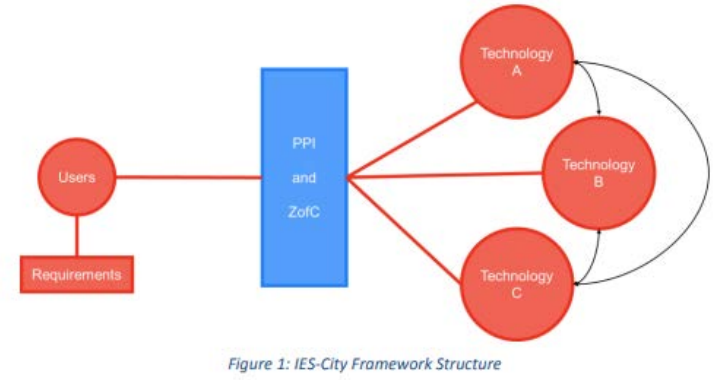


Figure 7.3-4 IES-City Framework[48]

Through these comparative studies, the concept of Pivotal Pointes of Interoperability (PPI) and their aggregate, Zones of Concern (ZoC) are formulated. The way of thinking behind them is that it is not always necessary to match the specifications of the entire system to ensure interoperability between IoT systems, but to adopt common technologies in critical areas as deemed appropriate. Examples of PPI include, as shown in Figure 7.3-5, IPv6 address of the federation section (Southbound Interface) which acquire data from data sources, REST API of the federation section (Northbound Interface) which provides data to applications above the data management & federation layer, etc.

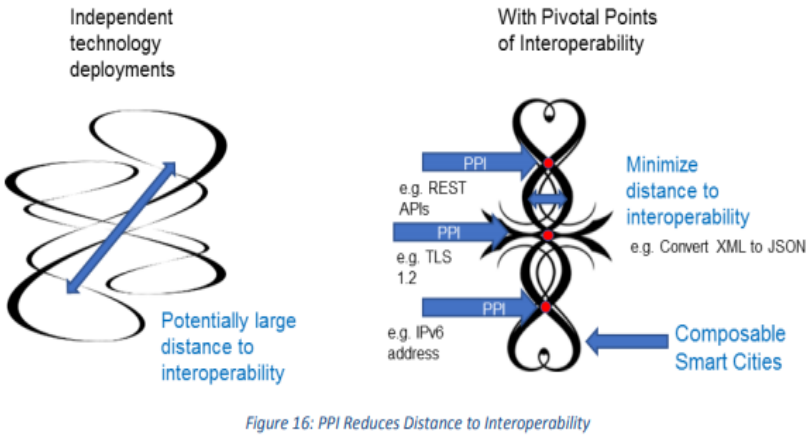


Figure 7.3-5 Pivotal Points of Interoperability[49]

[48] https://s3.amazonaws.com/nist-sgcps/smartcityframework/files/ies-city_framework/IES-CityFramework_Version_1_0_20180930.pdf P7

[49] https://s3.amazonaws.com/nist-sgcps/smartcityframework/files/ies-city_framework/IES-CityFramework_Version_1_0_20180930.pdf P71

7.3.1.3 Interoperability of City OS

Based on the concept of interoperability in Europe, it is understood that “Semantic Interoperability” and “Technical Interoperability” must align themselves with respect to external federation interoperability of City OS. The components of interoperability and the relationship between API provided on City OS and data model are defined as shown in Figure 7.3-6.

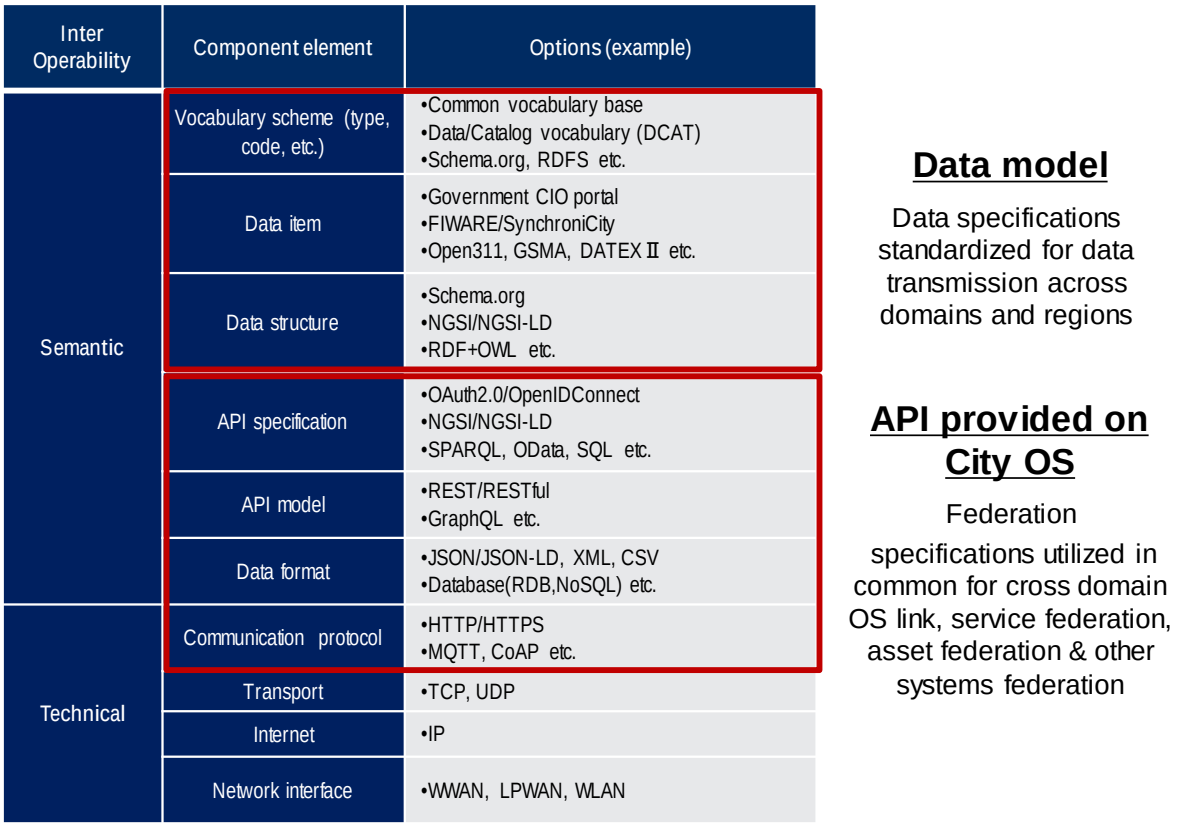


Figure 7.3-6 Consideration policies for API on City OS and data model

APIs implemented on City OS are described based on the following policies. For further details, please refer to “7.3.2 API and interface provided on City OS”.

- API describes functional requirements required for external federation, API specifications as standard specifications, and requirement categories.
- It is desirable to adopt APIs which conform to standardized specifications set by standardization groups.
- Data transmitted between City OSs should follow the movement of data model standardization in the future.

- APIs are subject to change due to continuous curation/development of the Smart City reference architecture.

Based on the consideration policies stated above, the implementation policies of City OS are described below.

(1) Actively adopt API, data model, etc. set by standardization body groups.

City OS is to be implemented with APIs meeting the functional requirements in accordance with the issues to be resolved and the goals of the region with reference to APIs provided on City OS. In doing so, interoperability can be easily ensured by actively adopting API specifications, data models, etc. set by standardization groups.

WebAPI is selected since APIs provided on City OS must be provided in the form which is easy to use for the users. WebAPI in City OS designates URI and manipulates resources by way of HTTPS protocol and standard data format based on REST API model. JSON data format is used for response data for the ease of machine-based interpretation. As for other technical matters to be considered, please refer to API Technical Guidebook[50] issued by National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariat.

For data models, recommended data set[51] promoted by National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariat, and a set of data standards and a code list[52] found in common vocabulary ground or Digital Government Standard Guidelines[53] can be utilized. Date/time, address, notation system for geographic coordinates, PoI (Point of Interest) code, open codes published by government ministries and agencies, etc. are available as references. These data models are promoted also by referencing schema.org as the de facto standard on the web. With respect to the data which cannot be covered by data models already mentioned, it is possible to refer to schema.org[54]. Additionally, for geo-space and infrastructure information, it is also effective to refer to data models presented in the

---

[50] Source: National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariat, API Technical Guidebook https://cio.go.jp/sites/default/files/uploads/documents/1020_api_tecnical_guidebook.pdf

[51] Source: Recommended data set promoted by National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariat https://cio.go.jp/policy-opendata

[52] https://imi.go.jp/goi/common/

[53] https://cio.go.jp/guides

[54] https://schema.org/

data platform[55] for land, infrastructure, transport and tourism promoted by Ministry of Land, Infrastructure, Transport and Tourism.

(2) Open access mechanism to allow access by various organizations

Easy to use environment of API and data model implemented on City OS is maintained for the users of City OS by publishing them as open API and open data by way of developers portal, etc. Users of City OS refer to published open API and open data, and implement external federation using identical format or by way of mechanical conversion.

(a) Publishing functionalities

- Open API (interface specifications, users guide, etc.)
- Open data (data item, data model such as data structure, etc.)
- Metadata (catalog data, organization/personnel, etc.)
- Terms of service (license, prohibited matter, disclaimer, etc.)

(b) Providing test and evaluation environment

- Developers portal (catalog, console)
- Sample code, library
- Sandbox

(c) Environment for information exchange

- Region

For more details on open access method for API and data model, please refer to API Implementation Guidebook[56] issued by National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariat.

[55] https://www5.cao.go.jp/keizai-shimon/kaigi/special/reform/wg6/20191105/pdf/shiryou2.pdf

[56] API Implementation Guidebook issued by National Strategy Office of Information and Communication Technology (IT), Cabinet Secretariathttps://cio.go.jp/sites/default/files/uploads/documents/1019_api_guidebook.pdf

7.3.2 API and interface provided on City OS

7.3.2.1 Authentication-related API

(1) Requirements for interface (API)

It is recommended to use OAuth2.0[57] in conjunction with OpenID Connect[58] (request approval of OAuth based on the specifications of OIDC). Table 7.3-3 defines functional requirements for use in external federation. In addition, specific examples of implementations are described in Appendix.

Table 7.3-3 Functional requirements for authentication-related API

No.	Individual function	Description	Mandatory
1	Authentication/ approval	Execute validation & issuing and disabling of access tokens utilizing qualification information (ID/password, biometric information, etc.) stored in ID management. Restrict specific usage capability based on the pre-registered user authorization. * Use of OAuth is recommended.	✓
2	Attributes acquisition	Acquire attributes information of the authenticated user. * Use of OpenID Connect is recommended.	✓
3	Personal authentication	In the case of authentication which requires high level of security such as the case using personal data, it is required to provide an authentication method which confirms the identity by multi-factor authentication with a combination of biometric authentication, individual number card, etc. The authentication of individuals may not be implemented on City OS but on each respective service.	–

[57] https://oauth.net/2/

[58] https://openid.net/connect/

(2) Standard specifications

Standard specifications related to authentication are described.

(a) OAuth（Open Authorization）

An approval framework which allows restricted access to HTTP services by third party applications. The latest standard is OAuth 2.0 published as RFC in 2012 (RFC6749, RFC6750).

(b) Open ID Connect

A version with an added simple identity layer on OAuth 2.0 protocol.

(c) SAML[59]（Security Assertion Markup Language）

XML-based standard specifications to enable user authentication and single sign-on across different internet domains. It was established in 2002 and updated as version 2.0 in 2005.

[59] https://www.oasis-open.org/committees/download.php/56776/sstc-saml-core-errata-2.0-wd-07.pdf

7.3.2.2 Data management-related API

As presented in ”7.1.1.2 Data exchange (flow)”, City OS manages various data of different characteristics. For external federation of City OS, it is important to facilitate data management and data access which make the best use of the characteristics of the managed data.

(1) Requirements for interface (API)

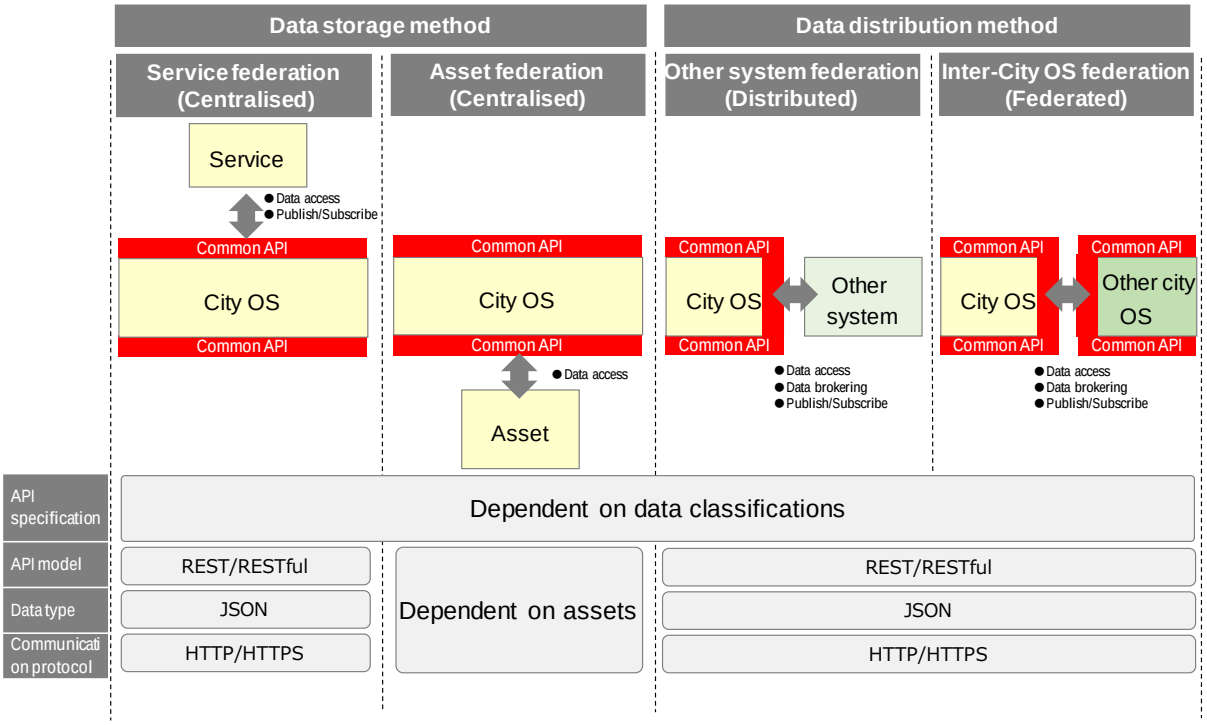
Table 7.3-4 Functional requirements for data access-related API

No.	Individual function	Description
1	Data access	Provide APIs to manage data lifecycle (register, refer, update, delete) incooperation with data management of City OS.
2	Publish/ Subscribe	Provide APIs to transmit a notice of change to notifying parties in real-time when the changes are made to the data stored on City OS. Also provide APIs to manage lifecycle (register, refer, update, delete) of the notices (conditions, notifying party, etc.).
3	Data brokering	Provide APIs to manage lifecycle (register, refer, update, delete) of the location of distributed data.
4	Personal data (sensitive personal information) delivery and acceptance	Provide this function in the case when personal data (sensitive personal information) is shared with Smart City services and other City OSs. It is required to confirm the identity prior to delivering personal data. As the method for confirmation of identity, multi-factor authentication with a combination of device authentication, biometric authentication, individual number card, etc. is to be used. Also provide functions to restrict the timeframe and the destination of data delivery. When the data is delivered, the usage log must be kept.

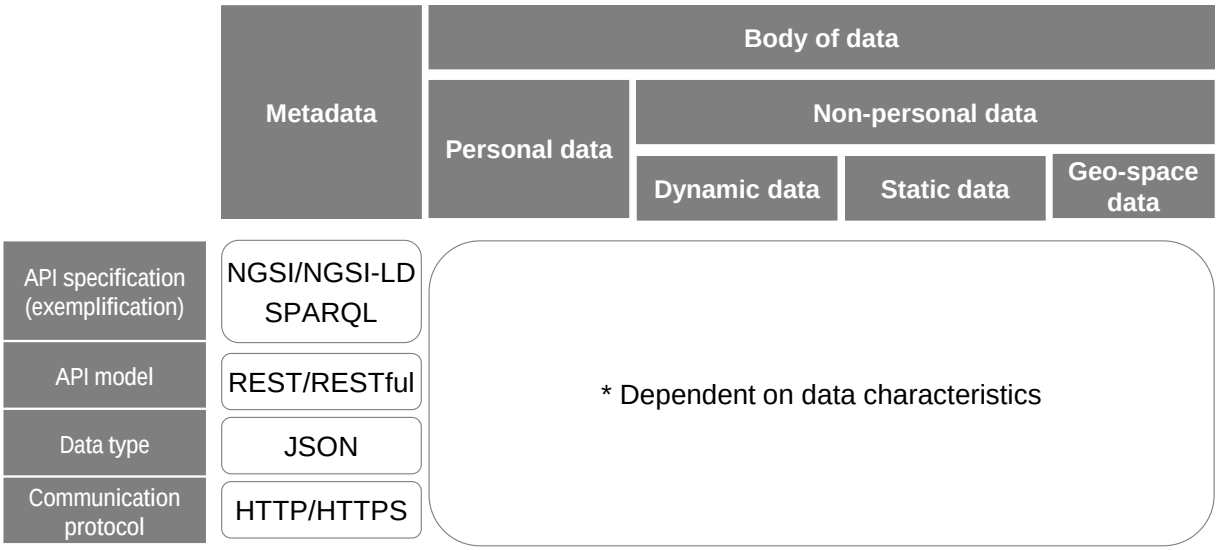
(2) Standard specifications

Standard specifications related to data access are described. Examples of standard specifications categorized by federation methods and data classifications are shown. The actual implementation may vary depending on the characteristics of the data managed by City OS. API specifications and data models adopted by City OS are made available as metadata for external access, and the users are required to access in accordance with the data characteristics. Specific examples of implementation by use cases are described in Appendix.

(a) By federation methods



(b) By data classifications



(i) NGSI[60]/NGSI-LD[61]

Acronym for Next Generation Service Interfaces. It was standardized by Open Mobile Alliance with follow-on updates, and the latest version, NGSI-LD (Linked Data) is published by ETSI (European Telecommunications Standards Institute). Physical objects in real world are managed by standardized data model as context containing unique identifiers, attributes, and other related supplemental information. Data exchange across domains and organizations are promoted via adoptions by FIWARE of the interface for the query of data location (NGSI-9) and the interface for the query of the data itself (NGSI-10).

(ii) SPARQL[62]

Acronym for Protocol and RDF Query Language. One of the query languages standardized by W3C. It enables designations of such basic query patterns as logical OR and logical AND, and also other patterns like text string manipulations and filters.

[60] NGSI https://www.openmobilealliance.org/release/NGSI/

[61] NGSI-LD https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.02.02_60/gs_CIM009v010202p.pdf

[62] SPARQL https://www.w3.org/TR/sparql11-overview/

(iii) REST/RESTful[63]

Acronym for Represented state transfer. It is one of the styles of software architectures for distributed hypermedia systems like a web. It is mainly comprised of the following four items of design principle.

- Stateless client/server protocol
- A set of “well-defined operations” applicable for all the information (resources)
- “Universal structure” to uniquely identify resources
- “Use of hypermedia” capable of handling both application information and status transitions

HTTP call interface for Web systems implemented in compliance with the “REST principles” is sometimes called “RESTful”.

(iv) JSON[64]

JavaScript Object Notation (JSON) is one of the text-based, light, and non-language-dependent data description languages. It is a format which is easy to read and wirte for humans and to generate parse for machines.

(v) HTTP/HTTPS[65]

Hypertext Transfer Protocol (abbreviation HTTP) is a communication protocol mainly used when a web browser communicate with web servers. HTTPS is an abbreviation of Hypertext Transfer Protocol Secure and an HTTP communication over secure connection provided by SSL/TLS protocols.

[63] Representational State Transfer (REST) https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[64] JavaScript Object Notation(JSON) https://tools.ietf.org/html/rfc8259

[65] HTTP https://www.w3.org/Protocols/

7.3.2.3 Service federation interface

(1) Requirements for interface

In service federation, common functions which improve user-friendliness are provided as user interface and APIs. There are few examples of those which can be described as standard specifications and they need to be improved in the future.

Table 7.3-5 Functional requirements for service federation API

No.	Individual function	Description
1	Service federation (payment transaction, etc.)	Provide functions to allow open access to APIs for services on City OS as APIs on City OS.
2	Regional point management	Provide functions to execute addition/subtraction/inquiry processing, etc. of regional points associated with users.
3	Opt-in management	Manage opt-in/opt-out options for the City OS users to determine which services are allowed to use the personal user information. It should be capable of managing the type of information to be provided. It should be capable of managing the opt-in/opt-out logs associated with the personal information transactions.
4	Catalog management	Execute registration/acquisition/search processing for the metadata stored in the catalog function (data catalog) of the developers portal site. * Reference: Basics for implementation of federation between data exchange platforms, published by Ministry of Internal Affairs and Communications.

7.3.2.4 Interface for federation with assets/other systems

(1) Requirements for interface

For Smart City assets and other systems which City OS federates with, it is necessary to federate to them by accommodating different data format, interface, communication method, and communication protocol.

Table 7.3-6 Functional requirements for interfaces for federation with Smart City assets/other systems

No.	Individual function	Description
1	One-way communication	Enable data access via common one-way communication protocol (HTTP/HTTPS). * For data access, please refer to data access-related API.
2	Bidirectional communication	Enable data access of and actuation to Smart City assets via common bidirectional communication protocol (MQTT, WebSocket, etc.)
3	Network interface	Network required for federation to Smart City assets varies in characteristics (communication distance, communication speed, power consumption, etc.) depending on the issues to be resolved or specifications of the connected devices. In addition to wide-area network (WAN) like 4G/5G, etc., low-power and wide-area network (LPWAN) such as the one used for IoT/M2M communications like LPWA, etc. should also be utilized.

(2) Standard specifications

Standard specifications for external data federation are described separately for federating with assets and other systems.

Table 7.3-7 Examples of standard specifications utilized in federation with assets /other systems

Inter-operability	Component	Asset federatoin	Other system federation
Semantic	Vocabulary scheme (type, code, etc.)	* Dependent on Smart City asset	* Dependent on system
	Data item		
	Data structure		
	API specification		
	API model		REST/RESTful, etc.
	Data Format		CSV, JSON, XML, WMS, Shape File, GeoJSON, Geographical Survey Institute tile format, XLSX(MS-EXCEL), Fude polygon, NetCDF, etc.
Technical	Communication protocol	HTTP/HTTPS MQTT, CoAP, etc.	HTTP/HTTPS, FTP/SFTP, SMTP(email), PubSub, PubSubPubPub, XMPP, etc.
	Transport	TCP, UDP	TCP, UDP
	Internet	IP	IP