# D1.3: Report and catalogue on the ICT data integration and interoperability

+CityxChange | Work Package 1, Task 1.2
Final delivery date: 30-04-2021


| | |
|---|---|
| **Deliverable version** | v.2.35 |
| **Dissemination level** | Public |
| **Authors** | Stephen Kinsella (UL);<br>Armin Shams (UL and Lero);<br>Markus Helfert (Lero);<br>Dirk Ahlers (NTNU);<br>Iyas Alloush (Lero);<br>Zohreh Pourzolfaghar (Lero);<br>Anthony Junior Bokolo (NTNU);<br>Sobah Abbas Petersen (NTNU) |
| **Contributors** | Michele Nati (IOTA), Nick Purshouse (IES),<br>Helena Fitzgerald (UL), Duncan Main (IOTA),<br>Mihai Bilauca (LCCC) |



This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 824260.

# Document Information

| | |
|---|---|
| **Project Acronym** | +CityxChange |
| **Project Title** | Positive City ExChange |
| **Project Coordinator** | Annemie Wyckmans, Norwegian University of Science and Technology |
| **Project Duration** | 1 November 2018 - 31 October 2023 |
| **Deliverable Number** | D1.3: Report and catalogue on the ICT data integration and interoperability |
| **Dissemination Level** | PU-Public |
| **License** | CC-BY4.0 Creative Commons Attribution, except where otherwise noted. https://creativecommons.org/licenses/by/4.0/ |
| **Status** | Complete |
| **Due Date** | 31-10-2020 |
| **Work Package** | WP1 – Integrated Planning and Design |
| **Lead Beneficiary** | UL |
| **Contributing Beneficiaries** | NTNU, LCCC, TK, IESRD, POW, FAC, TE, ABB, ATB, ABG, ESB, 4C, MPOWER, SE, COL, IOTA |

# Revision History

| Date | Version | Author | Substantive changes made |
|---|---|---|---|
| 20-10-2019 | v0.10 | Iyas Alloush, Zohreh Pourzolfaghar, Markus Helfert, Sobah Abbas Petersen | Initial document from discussions of T1.1/T1.2, separating topics between the two Deliverables |
| 20-12-2019 | v.0.16 | Armin Shams, Markus Helfert | Collaboration with partners; Recorded three API cases; Initial content on approaches/framework and DLT |
| 30-03-2020 | v.0.27 | Armin Shams, Markus Helfert | Cases and APIs expanded; Use and examination of guidelines/ standards/ frameworks started |
| 20-08-2020 | v.0.90 | Armin Shams, Markus Helfert | Section/subsection structure improved. Contents added on: API Cases, Requirements Table, API Catalog Enhancement, EA Use Case Template, Standards/Frameworks, References |

| | | | |
|---|---|---|---|
| 11-12-2020 | v.1.58 | Armin Shams, Markus Helfert, Dirk Ahlers, Anthony Junior Bokolo | Updates on Framework/Process for Evolving Solutions, Added FIWARE, other projects, further EU open data and vocabulary Resources and Standards, Automatic API Management Option, Advanced high-maturity option, content revised and harmonised after feedback from partners, WP lead, PM |
| 25-01-2021 | v1.94 | Armin Shams, Markus Helfert, Dirk Ahlers | Internal feedback round, merging and revising sections on background, method, approach, concise API Table added as Appendix, partner processes added |
| 28-02-2021 | v2.29 | Armin Shams, Stephen Kinsella, Markus Helfert, Dirk Ahlers, Sobah Abbas Petersen | EIF recommendations detailed; feedback integration, structure improvements; requirements table section re-written; merging and shortening; Standards/resources added to Appendix; API Catalog work restructured; practical info added. |
| 30-04-2021 | v2.35 | Stephen Kinsella, Dirk Ahlers, Armin Shams, Anthony Junior Bokolo, Markus Helfert, Sobah Abbas Petersen | Streamlining and restructuring of main content, catalog analysis, IOTA assessment added; overall streamlining, QA, finalisation |

# Table of Contents

# List of Acronyms

| | |
|---|---|
| **4C** | project partner |
| **ABB** | project partner |
| **ABG** | project partner |
| **API** | Application Programming Interface |
| **AtB** | project partner |
| **AWS** | Amazon Web Services |
| **DLT** | Distributed Ledger Technology |
| **EA** | Enterprise Architecture |
| **EAF** | Enterprise Architecture Framework |
| **eMaaS** | electric Mobility as a Service |
| **FAC** | project partner |
| **GDPR** | General Data Protection Regulation |
| **GTFS** | General Transit Feed Specification |
| **ICT** | Information Communication Technologies |
| **IES** | Integrated Environmental Solutions - partner |
| **IoT** | Internet of Things |
| **IOTA** | project partner |
| **KPI** | Key Performance Indicator |
| **LCCC** | project partner |
| **HAL** | API Manager |
| **HTML** | Hypertext Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **HTTPS** | HyperText Transfer Protocol Secure |
| **J2EE** | Java 2 Platform Enterprise Edition |
| **JS** | Javascript |
| **JSON** | JavaScript Object Notation |
| **JWT** | JSON Web Token |

| | |
|---|---|
| **MSP** | Managing Successful Programs |
| **MVC** | Model-View-Controller |
| **NeTEx** | Network and Timetable Exchange |
| **NTNU** | Norwegian University of Science and Technology |
| **ORM** | Object Relational Mapping |
| **ODM** | Object Document Mapper |
| **P2P** | Peer to Peer communication |
| **PMO** | Project Management Office |
| **PoC** | Proof of Concept |
| **POW** | project partner |
| **RA** | Reference Architecture |
| **RES** | Renewable Energy Source |
| **REST** | Representational State Transfer |
| **SDK** | Software Development Kit |
| **TLS** | Transport Layer Security |
| **UL** | University of Limerick |
| **URL** | Uniform Resource Locator |
| **URI** | Uniform Resource Identifier |
| **UML** | Unified Modeling Language |
| **VM** | Virtual Machine |
| **WP** | Work Package |
| **WS** | Web Socket |
| **XML** | eXtensible Markup Language |

# Executive Summary

This deliverable D1.3 is part of Task 1.2: 'Data Integration and Interoperability for the ICT Ecosystem' within WP1– 'Integrated Planning and Design' of the +CityxChange project.

The results presented in this report summarize the approach and achievements within the project to ensure data interoperability within a complex ICT ecosystem. The results have been developed in collaboration with partners and related ICT development tasks within the +CityxChange project. We have built on the Enterprise Architecture Framework (EAF) and ICT Ecosystem approach, detailed in Deliverable D1.2: Report on the Architecture for the ICT ecosystem. The results here specifically complement that enterprise architecture in order to ensure data integration and interoperability of ICT systems and services, including software platforms and tools, data repositories, and IoT devices. This report provides an overview of data integration challenges in the smart city domain including standards, data models, guidelines, APIs, etc.. The developed approach and lightweight data integration and interoperability framework DIIF aim to ensure agreement on open standards between service providers involved in demo projects for interoperability and support data flow between partners and demos. We document the interoperability work done between partners, the API management approach, and API catalog. To secure distributed data integration between various services in the ICT ecosystem, the potential of using IOTA and other Distributed Ledger Technologies (DLTs) as a mechanism for data transfer was examined.

This report addresses the data integration and interoperability goals of +CityxChange, through developing a lightweight +CityxChange Data Integration and Interoperability Framework. It specifically addressed two aspects (i) the Interoperability aspect under the Data Perspective, dealing with the general interoperability requirements and processes; and (ii) the DataxChange and Data Processing layers of the ecosystem, dealing with the overall data storage and exchange and its use in the data processing layer. The report concludes with a summary of lessons learnt that provide a valuable resource for replication within the +CityxChange project, application to other cities, and further research.

# 1. Introduction

This report is part of the +CityxChange Enterprise Architecture Framework (EAF) and ICT Ecosystem, detailed in D1.2: Report on the Architecture for the ICT ecosystem (Petersen et al., 2021) (see following figure). It is developed in collaboration with partners and the other ICT development tasks within the +CityxChange project.
This report addresses the *data integration and interoperability* goals of +CityxChange, through developing a lightweight +CityxChange Data Integration and Interoperability Framework. It specifically addressed two aspects of the EAF (D1.2), as highlighted in red:

- the Interoperability aspect under the Data Perspective, dealing with the general interoperability requirements and processes; and
- the DataxChange and Data Processing layers, dealing with the overall data storage and exchange and its use in the data processing layer.

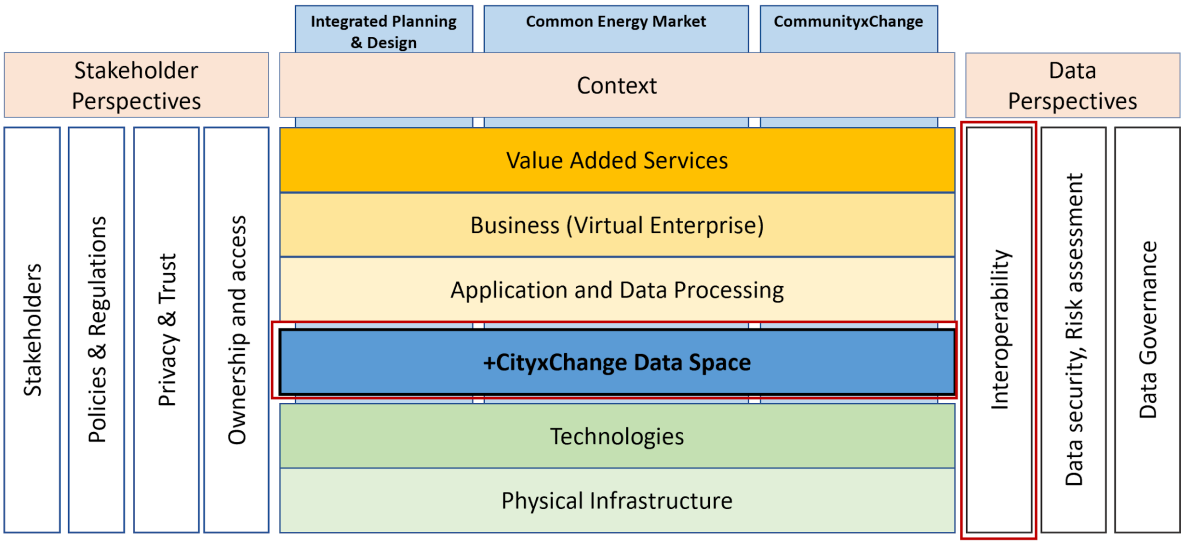


Figure:. +CityxChange Enterprise Architecture Framework (EAF), with the interoperability and data perspectives highlighted (Petersen et al., 2021)

+CityxChange follows the idea of a distributed ICT ecosystem for its data platform and computing needs. This is based on the complex nature of smart cities and smart city projects, where a number of stakeholders, including city administrations, come together. Instead of centralised platforms, the approach is to build open complex systems from distributed components so that actors are responsible for their own systems.
The collaboration and integration of these to build complex systems is achieved through an enterprise architecture approach as described in D1.2 (Petersen et al., 2021). It ensures that open standards and open APIs are used between partners for interoperability, following the ambition of the city as an open ecosystem (Ahlers et al., 2019) and city-as-a-lab and playground approaches (Wyckmans et al., 2019).

The rationale for the data integration and interoperability described in this report is the same as for the overall project Enterprise Architecture and ICT Ecosystem (D1.2). In a complex ecosystem such as a Smart City (and the demonstrators within +CityxChange), which uses loose coupling of components to achieve an open and distributed ecosystem, the autonomy of actors for their own areas/systems of responsibility should be ensured while strongly enabling the collaboration and integration between them.

There is a need for jointly agreed data standards, interoperability policies, and APIs. This ensures partners can interoperate, integrate and share data and applications with each other while keeping their own systems development separate. Cities have access to the data that they need for their roles, and open standards are used as much as possible to keep the ecosystem open and future-proof, as well as enabling access to other and new players as part of the Open Innovation strategy and approach (cf. e.g. D9.1 (Wyckmans et al., 2019) on collaboration, D3.6 (Fitzgerald et al., 2020) on Innovation Labs).

## 1.1 Context within the project

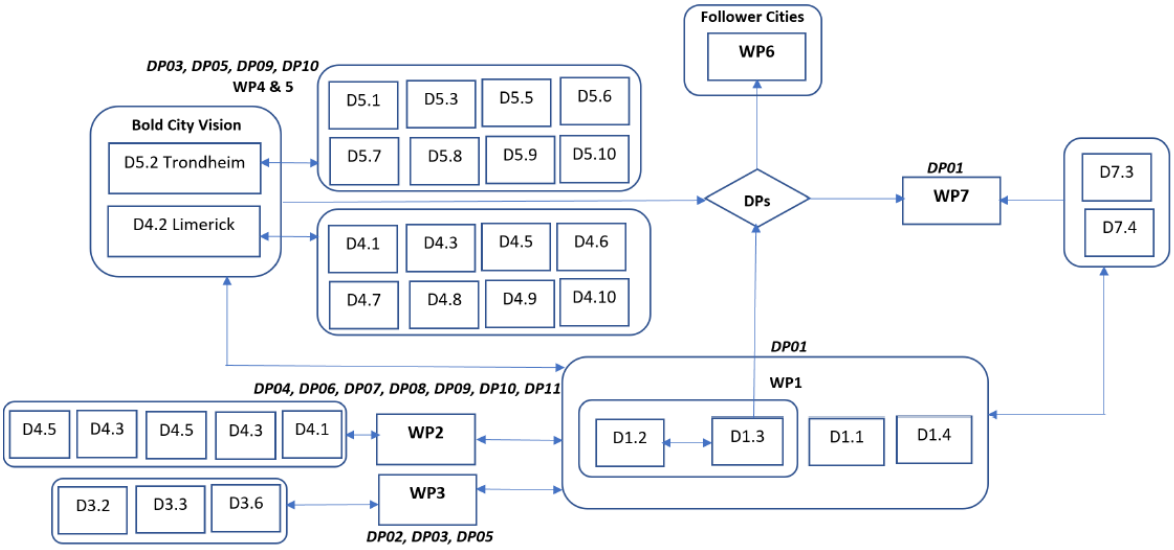


Figure: Context of this Deliverable within the overall +CityxChange structure (Source: D1.2)

Similar to the other tasks in WP1, this Deliverable D1.3 also relates to several other work packages in the project. As such, it combines information from several demonstration projects and the corresponding deliverables that describe the details of their technological solutions. The above figure shows an overview of the relationship between this deliverable D1.3 (of Task T1.2) and the other WPs and deliverables in the project.

## 1.2 Approach

We customise simple practical solutions for +CityxChange in its lightweight DIIF (Data Integration and Interoperability Framework, Sec. 3) through methods such as as co-creation of overall architecture, joint API Catalogue and a lightweight Cross-partner Data-related Requirements Table, based on ideas of organisational interoperability, knowledge management, and agile approaches.

This approach follows an agile approach in development, which is also suggested for the smart city context (Calzada, 2019; Faber et al., 2018) and is based on partners' needs from their other systems and deliverables. This task has taken an intermediate and facilitating role within the consortium, together with the ICT ecosystem tasks. The development and adaptation/deployment of individual systems happens throughout all work packages, and part of partners' effort in the underlying task was also to develop their own APIs and provide the interoperability of their systems. In that view, the co-development with partners has been a major aim to support their contributions to demo projects (as Virtual Enterprises, cf. D1.2). This is also suitable for the complexity of the smart cities field and the data integration and interoperability in this project. The practical focus is driven by the needs of the project partners and the wider ICT Ecosystem, leading to the selection of specific support and facilitation described here.

The following figure shows how the +CityxChange Data Integration and Interoperability Framework fits into the +CityxChange Enterprise Architecture Framework (D1.2) and shows its components. These are discussed in more detail in Section 4 and 5.

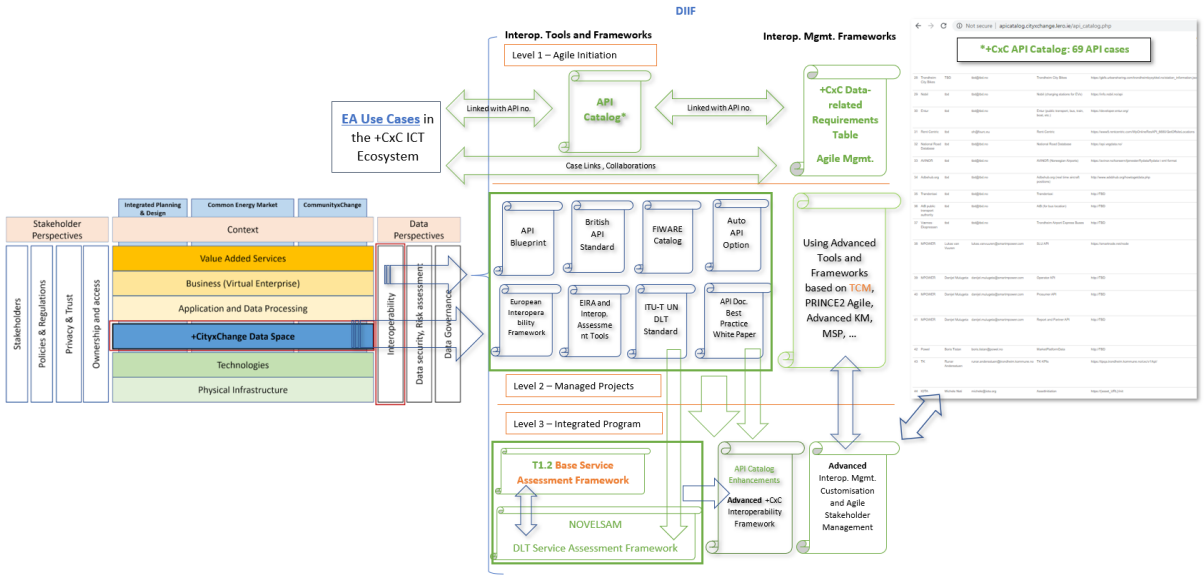


Figure: +CityxChange Data Integration and Interoperability Framework as part of overall +CityxChange Enterprise Architecture and Ecosystem

## 1.3 Objectives and Document Structure

This report addresses project needs as described in the Task description in the DoA (phrases in italics below) and how these are addressed in this Deliverable:

- *Ensuring data integration and interoperability of ICT systems and services* is achieved through the **interaction with partners** from WPs 1 (Integrated Planning and Design), 2 (Common Energy Market), and 3 (CommunityxChange), as well as the City and Monitoring & Evaluation WPs, as part of the overall development of the EA use cases of D1.2 and a range of separate data meetings. A lightweight **+CityxChange data integration and interoperability framework DIIF** has been developed and described in Section 3 and the resulting support artifacts such as the API catalogue in Section 4 and 5. It addresses this aspect as a specific topic within the overall +CityxChange Enterprise Architecture Framework (D1.2).
- *Identify open standards for data vocabularies and data models* in the smart city domain, as discussed in Section 4, with details presented in Annex F, and analysis of used APIs in Section 5.
- *Integration of API specifications*, already implemented API libraries, and partner APIs are linked to the point above and discussed in Section 3 and 4, and in Section 5 on API Catalogue and connection to EAF use cases.
- *Ensuring agreement on open standards* between service providers and supporting data flow between partners and towards the KPI collection into the monitoring platform in WP7. This has been achieved in the overall approach and framework, ongoing data meetings, in partners' mutual development of APIs, as part of the API Catalogue and use cases, in related other work and deliverables as summarised in Section 4, as well as the KPI collection discussed in Section 4.
- To secure distributed data integration between various services in the ICT ecosystem, the *potential of using IOTA* and Distributed Ledger Technologies (DLTs) as a *mechanism for data transfer* is examined in cooperation with WP2 in Section 6. Fall-back mechanisms in experimenting with DLT are briefly described there as well in collaboration with WP2 results.
- Section 7 discusses lessons learned and Section 8 presents conclusions.

# 2 Background: Data Integration and Interoperability for Smart Cities

Data Integration and Interoperability are vital for the creation of Smart City services. This is especially the case if services are composed of components from different vendors or stakeholders as is the case for +CityxChange, discussed in the use cases in D1.2. Data integration and interoperability between stakeholders can be achieved by using existing common data models or agreeing on the necessary data models to be exchanged through APIs in individual systems, depending on mutual requirements. APIs and data models provide standard vocabulary and communication channels, and help manage complexity. They can address important challenges from the need for federation to the silos challenge.

The aim of +CityxChange is to develop techniques and frameworks/processes that support sustainable positive energy approaches through the co-creation of Positive Energy Blocks with local stakeholders. This can be supported by improving the data interoperability, data integration, and exchange between the different partners and stakeholders. Supporting these topics through lightweight frameworks and processes, introduction of best practices and recommendations is a major contribution for this task.

## 2.1 Smart City data

Smart Cities are a result of taking a multi-faceted approach, including numerous stakeholders orchestrated towards the aim of improved urban experience, quality of life, sustainability, liveability, and service enhancement, to name a few, through *technology*, while the *people* and *process* aspects also play key roles (Ahlers et al., 2019; Pimenta de Miranda, 2019). In particular, a Smart City in the ICT view is a city where ICT is heavily used for enhancement of the services provided to citizens and for creation of value. Innovation and sustainability are also among the key aspects for Smart Cities. “Smart City” (Ojo et al., 2015) “is an interplay among technological innovation, organizational innovation, and policy innovation”. From one point of view, there are at least six main sectors that are impacted by smart city initiatives (Alawadhi et al., 2012): environment, transportation, energy, governance, people and communities (Citizen participation, lifestyle and health can be listed under this category), technology, and built infrastructure (e.g. roads).

### 2.1.1 Data Barriers and the silo challenge in Smart Cities

In the context of smart cities, data plays a major role in both development and functionality of services.
Using scattered databases is a complex process due to the challenges, including the variety of technologies and data models involved. Data sources are plenty and data collection

devices/sensors are heterogeneous and produced by different vendors. This creates difficulty to link and aggregate the data collections in order to provide a reusable form of published data that can be used easily through different applications for smart cities (Bhatt et al., 2017). In addition, city data is often historically organised along department and sector lines, with limited cross-use even within municipalities, and less to outside stakeholders. This is known as the "silos challenge" (as illustrated in the following figure).

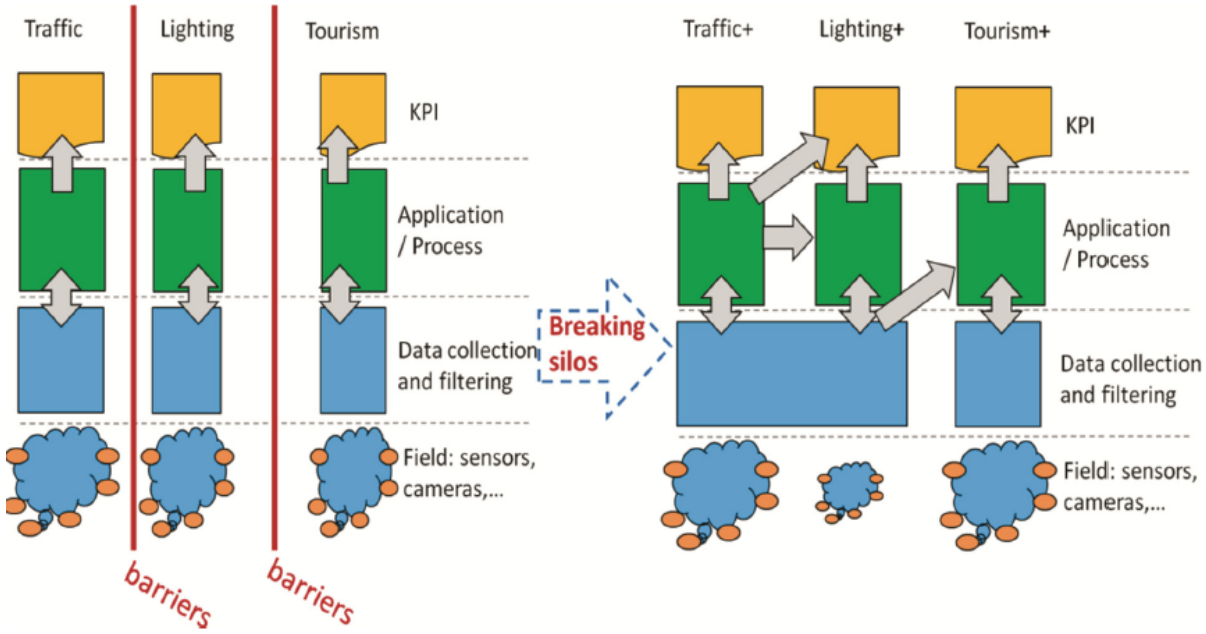


Figure: The silos challenge, among the barriers to creating smart cities (source: (Bhatt et al., 2017))

## 2.2 Interoperability

*Interoperability* is defined for example by IEEE (IEEE dictionary contributors, 1990) as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged". In Open Group's TOGAF 9 (basis for the +CityxChange EAF), interoperability is defined as the ability to share information and services (The Open Group, 2011).

*Integration* aims to connect different systems towards providing joint functions as an aggregation. System integration can be defined as practices for bringing together elements or subsystems. Data aspects are then covered by data integration.

The large amount of data and cross organizational interactions between different businesses lead to complex models of *data exchange interactions* with various types of interoperability and usefulness (van der Aalst, 1999). Communication processes within a collaborative process can have a great range of formats, structures and contents (Ziemann, 2007).

Often interoperability is considered as a technical issue, involving syntax and semantics of the data that is transferred or shared. Several approaches have been described in the literature to achieve technical interoperability, some of which include agreed standards, data formats, or federated approaches. The challenge of ensuring technical interoperability and integration can be understood under the topic of organisational interoperability. An organisation as a system needs higher maturity in order to better benefit from data and be able to create value and to progress in its interoperability dimension. Systemic effects of maturity affect all the organisation aspects, including data interoperability and integration. Since the project consortium is not a single organisation, approaches need to take the distributed governance into account. Maturity concerns are therefore more focused on the interactions between partners and systems, leading to the approach followed in this work. Achieving organisational interoperability requires addressing interoperability at many levels, beyond the technical level. This is illustrated in the figure below, which identifies the need for process, knowledge, value and goal interoperability, to achieve wider complete organisational interoperability.

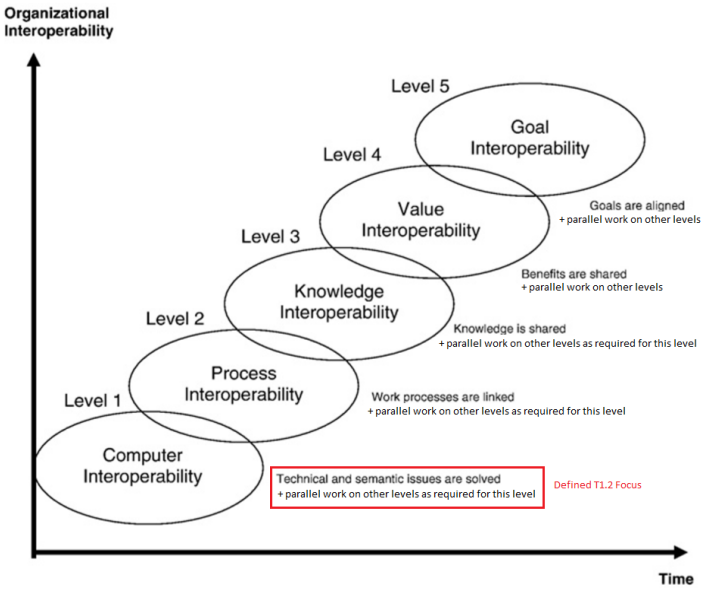


Figure: Organisational Interoperability Diagram, highlighting its level 1 on the technical interoperability as the main aspect for the DIIF developed in this report (adapted from (Gottschalk, 2009))

This high complexity challenges enterprises when they need to design their business processes. The inherent complexity of interoperating organizations can often not be removed or reduced, but it can be managed (Ziemann, 2007). Modelling and describing systems relying on well-organized architectures can help to manage this type of complexity such as in Enterprise Architecture (Achilleos et al., 2008; Chiprianov et al., 2011; Chiprianov et al., 2011b; Petersen et al., 2021).

Interoperability has also been the focus of several European projects, such as ATHENA and the European Network of Excellence INTEROP[1], resulting in frameworks that could support interoperability among organisations. The ATHENA (Berre et al., 2007) Interoperability Framework (AIF) focussed on achieving model-based collaborative work across organisations by achieving information interoperability, flexible composition and execution of services and cross-organisational business processes (Berre et al., 2007). The emphasis here is on achieving collaborative business processes across organisations. The European Interoperability Framework (EIF) enhances beyond organisational interoperability to include additional layers, which support the digitisation of public sector services, governance and interoperability across organisations and borders. The six layers included in EIF are (EIF Contributors, 2020):

1. Technical Interoperability
2. Semantic Interoperability
3. Organisational Interoperability
4. Legal interoperability
5. Integrated public services governance
6. Interoperability governance

We take up part of this in the development of the methodology in Section 3 and 4. Further details are found in Annex G.

The European Interoperability Framework for Smart Cities and Communities (EIF4SCC) is also being prepared and reported[2] on the Joinup network at the time of writing. Consultations with experts are ongoing on EIF4SCC. As it is derived from EIF, it covers Technical Interoperability besides other wider aspects such as Organisational Interoperability, and aspects such as Cultural and Legal Interoperability.

## 2.3 Application Programming Interfaces (API)

Data and functionality is exchanged between smart city stakeholders according to organisational missions and business models. Every stakeholder has a specific role according to the provision model that is used. One of the aims of this D1.3 report is to highlight the importance of using APIs for the exchange of different types and categories of data and functionality, especially within distributed systems for the demonstrations (and to address challenges such as the data silos challenge).

The objective of APIs (Application Programming Interfaces) is to provide a formal way to describe the way different systems can communicate in order to achieve interoperability.

[1] http://interop-vlab.eu/interop/

[2] https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/news/eif4scc-smart-cities-communities

APIs are now the preferred way to structure distributed web-based or internet-mediated systems (Medjaoui et al., 2018; Amundsen, 2020). They are a necessity for microservice architectures, which we do not follow up on here. APIs expose specific functionality including data access to systems, while hiding most of the complexity and implementation details of the underlying platforms and databases. They play a mediation role by bridging the gap between the requests of clients and the different distributed data sources. Moreover, APIs can provide data access mechanisms that organise and secure the access to the data. In short, an API exposes functionality, while the data exchanged is described by data models.

For enabling interoperability and effective API development, the diversity, heterogeneity, and autonomy of software components, application solutions, business processes,business value, and the business context of an enterprise must be considered (Berre et al., 2007). Cross-organizational processes today usually rely on web services to exchange data between systems and services (Saleem et al., 2014). While these are valuable goals, implementation can have its challenges and they are not a solve-all.[3]

API development frameworks rely on an adapted client-server paradigm. Behind the API as an exposed endpoint, internal system details are encapsulated and abstracted from the clients. In most cases, APIs connect to internal databases for data storage or processing. There are 3 main types of APIs in Web-mediated services today: REST (and increasingly GraphQL), SOAP, and WebSockets. REST and SOAP fully use HTTP methods while WebSockets build its implementations on top of the TCP protocol (Fette & Melnikov, 2011). REST is the most used protocol with a simple semantic, often supported by the JavaScript Object Notation (JSON) format to encapsulate exchanged data. Remote Procedure Call (RPC) provides another way to build communication between API endpoints through HTTP but is mainly used in legacy systems.

There are many frameworks that can be used to build web applications such as Symfony, React/Meteor, NodeJs, and Prisma. While their objective is similar, they are based on different languages or their database support. Frameworks for communication between nodes in a cluster are out of scope here. Since APIs abstract away from these internal implementation details and focus on the communication part, it is sufficient to have an agreed specification of the API protocol, data format, and data model, while partners are free in their internal implementation.

Questions of Data Interoperability and Integration Standards are discussed in the remainder of this report, as part of methodology, the API use cases, and the guidelines.

[3] https://nordicapis.com/five-common-misconceptions-about-apis/

Detailed explanations of standards are found in the annex. Specifically used standards are discussed within the API cases or in the catalogue.

## 2.4 Open Data

Open data can be defined as "data that can be freely used, re-used and redistributed by anyone - subject only, at most, to the requirement to attribute and sharealike"[4]. This means that the access is open, but certain selected limitations may apply. A precise meaning of "open" with respect to knowledge is available: "Knowledge is open if anyone is free to access, use, modify, and share it — subject, at most, to measures that preserve provenance and openness"[5]. According to the EU commission's Directive 2003/98/EC[6] for Open Data Strategy, the reason behind publishing Open Data is to allow the reuse of public data by private companies and organizations.

In the context of +CityxChange, and in correspondence to D1.1 and D1.2 (WP1), there is a need to reuse processed data to create new added-value services and evaluate KPIs. Furthermore, open data enables open innovation and engagement of residents and stakeholders in addressing city challenges (Ojo et al., 2015). The +CityxChange Data Management Plan (DMP) (Ahlers et al., 2020) explains the use of Open Data in more detail, especially the project ambition under H2020 to make data available under FAIR (Findability, Accessibility, Interoperability, and Reusability) principles[7] for reuse.

## 2.5 Data Governance

Data governance is one of the cross-sectoral topics of the +CityxChange EA.In particular, "Data Governance ensures key data management processes and encompasses people, organisations and processes. Data governance also ensures that data is consistent, available and usable." (Petersen et al., 2020) It includes aspects of the quality, usability, availability, security and consistency of an organization's data. It can be understood as a high-level organisation data management aspect and a micro-scale level of individual data sets and (shared) data models. While the former is covered within the overall Enterprise Architecture (EA) for +CityxChange as a whole (cf. D1.2 (Petersen et al., 2021)), the latter is handled in more detail in this document. Further details are also found in the project's Data Management Plan (DMP) D11.16 (Ahlers et al., 2020). It focuses on data governance from

[4] https://opendatahandbook.org/guide/en/what-is-open-data/
[5] https://opendefinition.org/od/2.1/en/
[6] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2003:345:0090:0096:EN:PDF
[7] European Commission, H2020 Programme – Guidelines on FAIR Data Management in Horizon 2020
https://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/hi/oa_pilot/h2020-hi-oa-data-mgt_en.pdf

the management view of the project and specific requirements under H2020 on open data, FAIR principles (see previous section), data management and access, ownership, licenses, consent, GDPR, etc. through regular updates.and general requirements of the project in line with H2020 principles.

# 3 Methodology for Data Integration and Interoperability Framework for +CityxChange

This section describes the +CityxChange Data Integration and Interoperability Framework (DIIF) and the methodology of its development. It extends the +CityxChange Enterprise Architecture as described in D1.2: Report on the architecture for the ICT ecosystem, and elaborates the interoperability aspect (vertical Data Perspective) and the DataxChange (Data Space) layers. This is explained in further detail in Section 5 for the ecosystem use cases, which are central as views of the project Demonstrators.

## 3.1 Methodology

Here we describe the overall approach to develop the DIIF as a main outcome.

The main +CityxChange work follows an iterative and agile approach, as does the work on data integration and interoperability. As the designs, APIs and collaborations are developed and implemented by different partners, the work needs to accommodate the different paces of the consortium in developing systems, absorbing frameworks and best practices, and evolve the interoperability, integration, and interfaces. Thus, an agile method is advised for data integration and interoperability management, which is in line with other Smart City approaches (Calzada, 2019).

In order to develop the specifics for the DIIF we follow a hybrid methodology approach that is based on Design Science (Johannesson & Perjons, 2014; Järvinen, 2005; Hevner, 2007) and Action Research (Järvinen, 2005). It includes different best practices and standards according to partner needs, the needs understood through data meetings and further studies, into the DIIF design. *Action Research* as a method, focuses on problem solving through social and organizational change, while the *Design Science* paradigm and underlying methods focus on problem solving by creating and positioning a design artifact in a natural setting, or exposing it to expert feedback where practical experiments cannot be done. Action research does discovery-through-action and design science does discovery-through-design (Baskerville, 2008). The methodology entails zooming in on specific parts of the literature and coming up with innovations (e.g. in level 1 and level 3 of our framework), the criteria being partner feedback during specific meetings and expertise of the internal team. This creates the *first framework design* which can be evolved using a *Design Science Methodology* (together with internal/external experts and by conducting formal survey iterations to evolve the design).

## 3.2 Data Integration Interoperability Framework (DIIF)

Within a smart city, it is paramount that people, process and technological means can be orchestrated towards common smart city goals such as sustainable creation of value. Therefore high maturity of *organisational interoperability and integration* are needed starting from the technological interoperability and considering data/service semantics aspect. This establishes a mutual conceptual and technological ground for collaboration and interoperability between stakeholders. The DIIF integrates different best practice views relevant to data integration, interoperability and interoperability management, with direct artifacts and innovative outcomes, as well as suggesting process and management best practices which help ensure data integration and interoperability.

One example of a customised easy-to-use solution is the API Catalogue (at a basic level 1), while its future extension ideas introduced in this document is based on a level 2 best practice and falls itself into level 3 (direct extension on the existing catalogue and eventually using the automatic API Management options). Another example is focusing on some of the concepts such as Organisational Interoperability, KM, PMO, PRINCE2Agile and MSP (Office of Government Commerce GB, 2007) which we have shrunk to an innovative light-weight Cross-partner Data-related Requirements Table. This is the most lightweight alternative we know for the mentioned best practices (PRINCE2Agile/MSP), concepts (Organisational Interoperability/KM) and organisation parts (PMO). This oversimplification is only advised for the first steps of the Agile development where there are also resource constraints.

Similar to D1.2, the focus here was on practical support of the Demo Projects and the facilitation of interoperability processes. Further formalisations of the framework and of experiences are planned through future scientific publications.

The DIIF ensures the support and facilitation of the processes towards the implementation of the +CityxChange Demo Projects, from the data integration and interoperability point of view in the ICT +CityxChange ecosystem. Many of the discussions are incorporated throughout +CityxChange within tasks; we refer to the relevant other Deliverables with these results in Section 4 and 5. Details are discussed there and in the conclusion.

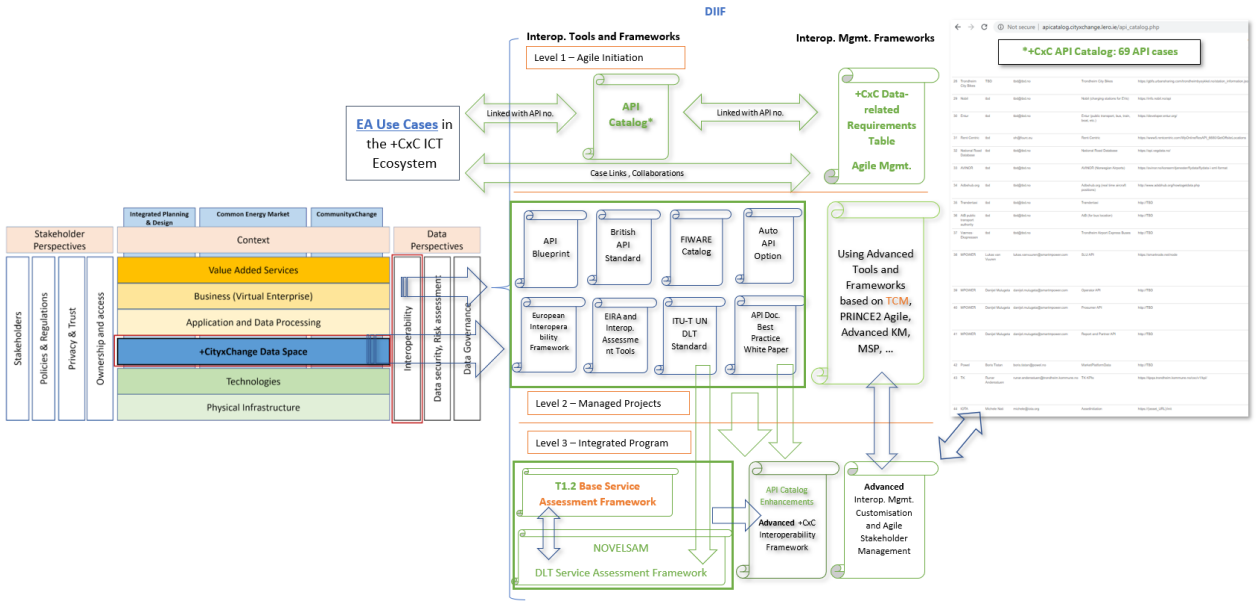


Figure: +CityxChange Data Integration and Interoperability Framework (DIIF)

## DIIF Multi-level framework

The DIIF (Data Integration and Interoperability Framework) is structured into 3 layers as a process for Evolving Interoperable Solutions for +CityxChange. As a framework, it is aligned with the work of D1.2, and co-designs the direct practical steps.

**Level 1** extends the ICT development process supporting the overall development of ICT systems and the Enterprise Architecture (EA) across the Ecosystem. This level includes the use of an API catalogue directly used in the EA use cases of D1.2; and puts forward a cross-partner data-related requirements table which facilitates the process of recording and brokering data-related requirements between partners. Level 1 is more concerned with computer and technical interoperability as referred to in the organisational interoperability literature. Subsequent levels are designed to focus more on the complexity of the organisational, business and governance needs for interoperability, thus capturing more of the organisational interoperability.

The agile development at Level 1 is done together with the ICT partners and cities within +CityxChange, leading to internal living documents and collections being used as valuable resources/tools supporting the integration tasks within +CityxChange. They represent initial and informative steps on collaborations for data interoperability and integration. This represents the main part of the work done in collaboration with D1.2 as shown in the overview and the following figure.

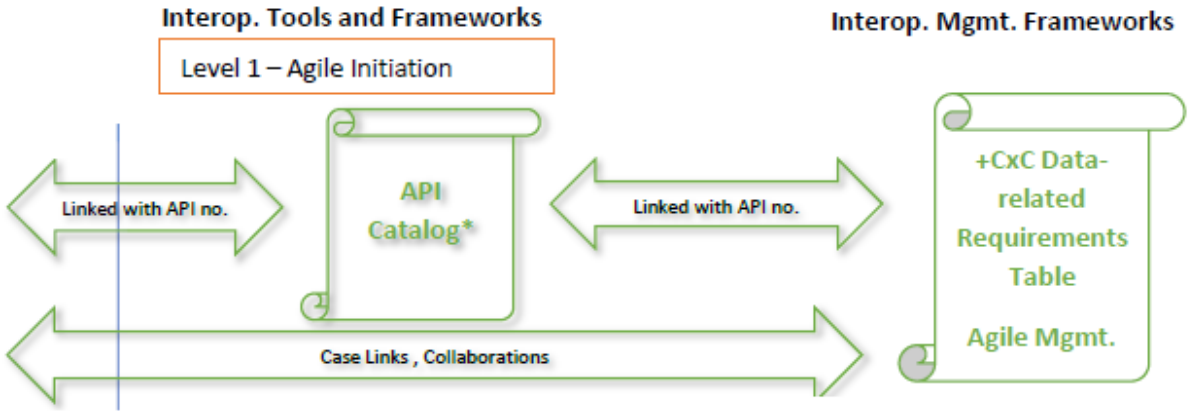


Figure: Lightweight tools for +CityxChange for starting interoperability and integration

**Level 2** extends the development process in Level 1 by providing further selected resources such as best practices and standards. Related rules and examples are extracted with the aim of increasing quality of development, maintainability and fostering better development and integration processes. This layer provides selected support and guidance for the development, documentation and management of APIs as shown in the following figure.

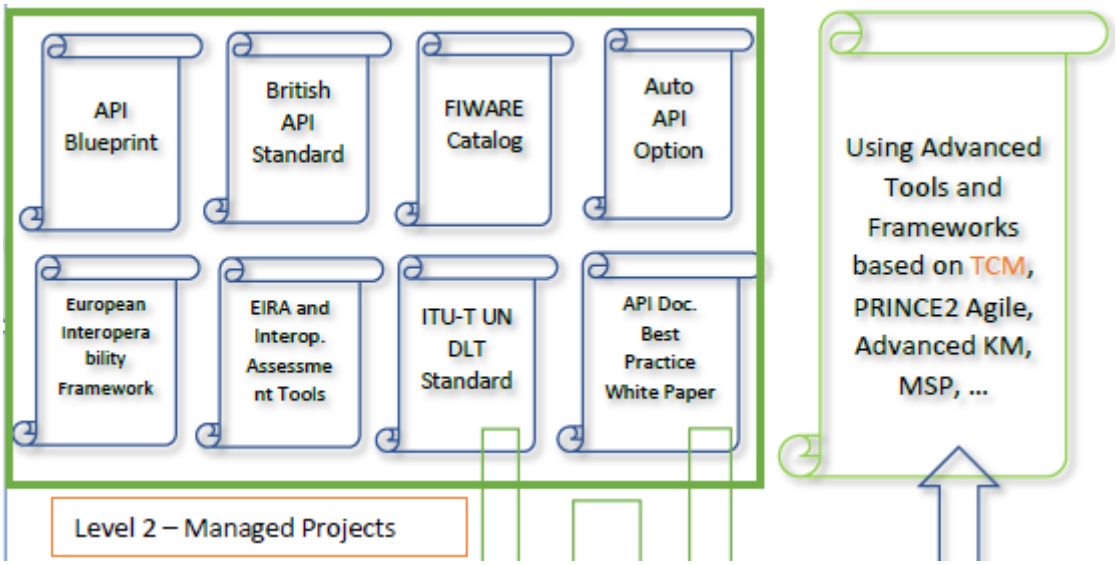


Figure: Selected resources for API improvements

**Level 3** focuses on customisation of the level 2 selected resources, suggesting innovations on top (such as NovelSAM), and achieving automation and optimization that includes for example automatic API publishing, discovery, monitoring and management. Customisations and maturity assessments can form a unified blueprint. Initial work was done with partners to demonstrate customisations of best practices, API Catalogue enhancements, and lessons learned after 2 years of these processes as seen in the following figure. This level is left mainly for future work.

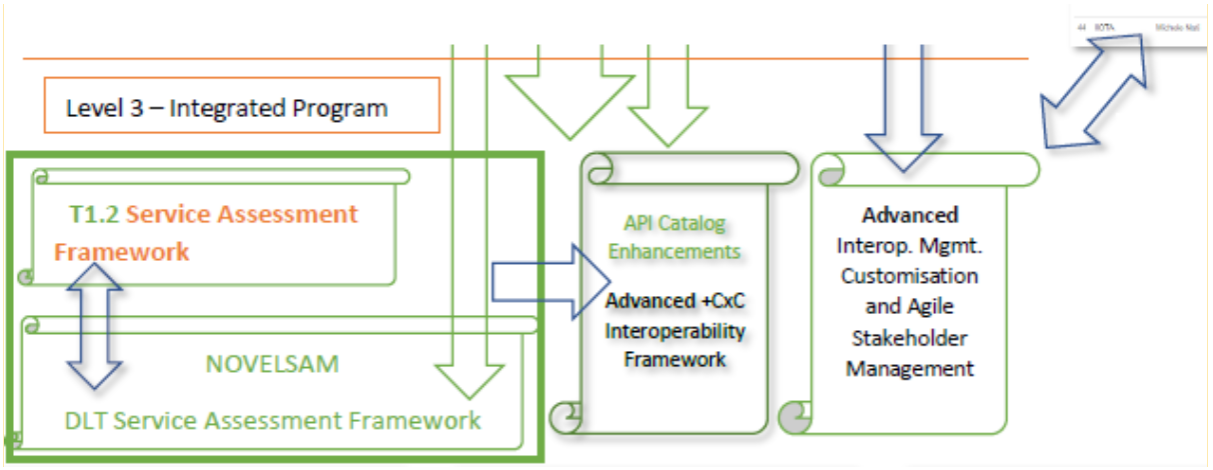


Figure: Advancing maturity based on best practices and experiences

# 4 Ensuring Data Integration and Interoperability: process & artifacts

## 4.1 The RI&D Process as executed so far

In order to ensure interoperability and data integration some form of *interoperability management* is required. During the course of the RI&D and feedback with the project a requirement for an easy-to-use and simple tool/framework for the consortium for management of progress in data interoperability and data integration became evident.

In the early phase of the project we obtained an overview of relevant data that the cities had and to understand more about the data; e.g. the types and formats of data, ownership and accessibility (both technological and role-based access), if there were APIs to access the data, how frequently the data were updated, etc. An overview of the information that was gathered is shown in the following Table. The aims of the table is to provide examples of some data and to highlight the diversity of the data. A synthesis of the data highlights the variety of data that is relevant to the +CityxChange project. This sample of data enables us to highlight some of the issues that have informed the design of principles and guidelines for the project and the interoperability framework.

- There is a variety of data and data types.
- Data is owned by multiple parties.
- The relevant data includes personal data.
- Data is stored in a variety of ways (e.g. files, databases) and stored in multiple places (e.g. private and public clouds).
- The data models for all data are not known.
- The data is updated at different frequencies (e.g. hourly, daily or as needed).
- The reliability of the data varied from High to Low or are sometimes unknown.
- Some of the data has an API access, but not all.
- The degree of openness of the data varied from closed to opendata.

| | Cities | | | | | | |
|---|---|---|---|---|---|---|---|
| Aspects of data | Trondheim | Limerick | Smolyan | Voru | Sestao | Pisek | Alba Iulia |
| Examples of types of data available in each | Energy measurements, air quality data | Live traffic data, city development plan | Register of all buildings, municipal budget | Information about buildings, information about | | Registry of land units and energy consumption in | Route network data, Electronic payments |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| city that could be used in services | | | | land usage | | Municipal buildings | |
| Who owns the data? | Municipality, 3rd party, Open data | Municipality, 3rd party, Open data | State and Municipality | State and Municipality | | Municipality and 3rd Party | Municipality mostly |
| Do they have personal data? | Yes | No | Yes | Yes | | Yes | Yes |
| Data format? | Varied, mostly xls/CSV | CSV and Geo-JSON | Excel and DB | DB | | DB and PDF | DB, HTML and CSV |
| How are they stored? | DB | File, DB? | File and DB | DB | | | DB |
| Where are they stored? | Public & private cloud | Unknown | Always internal server | Public & Private cloud, local pc | | Private and Public clouds,internal server | Private and Public clouds,internal server |
| Is the data model known? | Mostly unknown | Unknown | Unknown | Unknown | | Unknown | Unknown |
| Frequency of data update? | Hourly, daily, monthly, yearly, realtime | Mostly - as needed | Realtime, daily, as necessary | Daily, Monthly, as needed | | Realtime, daily, yearly, as necessary | Realtime, daily |
| Reliability of data? | Varied (high, medium, low) | Unknown | Medium and High | High | | High and Low | High and Medium |
| Degree of openness / accessibility | Open, group and specific access | Mostly - Opened | Open and Specific access | Open and Specific access | | Mostly open data | Open and Specific access |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| API access? | Some | No | Some | Most of them | | Some | Some |

*Table: Initial data understanding of the participating cities*

In order to ensure Data Integration and Interoperability we developed and introduced an innovative light-weight tool: The Cross-partner Data-related Requirements Table, which was encouraged and details communicated in one-to-one meetings with partners and in broader workshops. Information has been ongoing provided by partners. During the project we also started to advise on other tools and frameworks based on best practices from the beginning. However, in order to allow flexibility, an agile approach was confirmed early in the project, involving starting with a light weight management tool. The Cross-partner Data-related Requirements table helps to ensure data integration and interoperability of ICT systems and services, including software platforms and tools, data repositories, and IoT devices (based on description T1.2). It helps with fast understanding of +CityxChange ICT ecosystem, receiving expert and partner feedback and reproducibility of the results in follower cities.
The requirements table is a matrix of demands, offers, and responses between these for the APIs and data exchange needs between partners. It takes a brokerage function for data-related requirements together with the facilitation that is happening in the data meetings and in discussions between partners.

Rows of the table: asking partner (partner requesting something to be done)
Columns: asked partner (regarding doing the request)
Table Cell Composition (can be text, or a file-link with password):

1. Data requirement elements (which asking partner wants from the asked partner): this can be API/data in different forms and/or the quality of such data/interface (hand-prepared data required by partners from each other for "integration and interoperability" itself can also be mentioned)
2. EA case link / software system name / API name concerned for each element (EA case link: fast understanding and access to details in a drill-down way, shall include API codes linked to the API Catalogue)
3. Requirement justification for the system/API (not needed if an EA case has been introduced above, though)
4. Fulfilment progress (estimated % preferred, or: NotStarted/Low/Mid/High/Done)

The present state of the table is available internally. Below shows an example of the anonymised version of entries. (Note: the italic parts are not part of the core table, the cell shows what anonymised partner X wants from partner Y, the cell info is real):

| | Partner X | Partner Y | ... | ... |
|---|---|---|---|---|
| Partner X | | **Required:** Data Model and formats regarding *system*Y data<br><br>**System:** systemX<br><br>**Related Case:** Limerick Case *(linked to the relevant EA use case)*<br><br>**Justification:** design and provide APIs to store them onto Ledger to guarantee their integrity and immutability<br><br>**Progress:** Done. *(systemX API-Z recorded in the catalog and reflected in the use case and communicated to the partner and solved the due requirements they were interested in)* | | |
| Partner Y | | | | |
| ... | | | | |

*Table: Example view of the cross-partner data requirements table with an anonymised entry, showing the start of a discussion on API needs*

We demonstrated that the requirements table reduced communication barriers for collaboration regarding achieving data integration and interoperability. We communicated the table to the councils of Limerick and Trondheim and it was received well. We could demonstrate that the requirements table could solve partner problems, for example when partners were not already working closely within a task or some other communication issue between the technical experts was present.

## 4.2 Reference to other interoperability Frameworks

In section 2.4.4, we described how the criteria for development and comparison of DIIF emerged. The mentioned criteria was evolved as a result of RI&D together with the project partners. We have built our work with two relevant frameworks - the EIF (European Interoperability Framework) (EIF Contributors, 2020) and AIF (ATHENA Interoperability Framework) (Berre et al., 2007), availing of the Methodology, Technical Data Integration Standard, Standards for Federated Technical Interoperability, Reference Architecture

and Interoperability Assessment Tool. This has been expanded with Agile Methodology and some initial tools to support our integration work providing some best practices within the project. Furthermore our DIIF includes non-technical (Organisational) Interoperability aspects and an implementation process suitable for the project, which is intended to allow replication among the project partners.

## 4.3 Integration of Specifications: API Catalogue Design and process

API Integration appears in two senses in the literature, one sense points to data integration *using* APIs, the other is integration of various API specifications under unified API Catalogue and eventually under an automatic API Management system. The API questionnaire communicated to partners in order to achieve agreement on a common specification format is the basis of our API Catalogue (as the current practical design, before possible automated options later). It is used to record partner API information. Subsequently, an enhancement of this design has been introduced and is being discussed, but has not been implemented yet at the time of writing this report, since it was not part of the partners' prioritisations in their demos.

### 4.3.1 Process & Interaction for Documenting API use

One of the aims of this task is to compile a register of APIs so that partners have an overview of available data and an API to access the data. We believe this to be a continuing task over the next two to three years. Hence, we developed a questionnaire designed for gathering information about available APIs and which data could be accessed through them. The questionnaire is designed based on the details that are required for a user to know about and use the API, from API purpose and description to the owner firm, contact person and API address, to qualities and characteristics such as being open or not, and the API specification. The questionnaire was initially shared among the partners using a Google spreadsheet (shown in the following figure), so that all partners could add their APIs and relevant information to the shared spreadsheet. An overview of the type of information to be gathered through the questionnaire are:

- Contact details for the API provider.
- About the API, e.g. version, objectives.
- About the data that could be accessed: the inputs and outputs.
- Protocol used for the API.
- The openness of the API and the data accessible through the API.

The process included: literature and system study (see D1.2 and Annex F) and the development of the catalogue structure; partner agreement on the specifications;

implementation of actual catalogue; interaction with and between partners to provide API specifications (see following Section for details on the results).

## 4.4 Examples of data models in other project work

In this section, we briefly point out additional examples of interface work done and completed so far in the early report. Further work, especially on the local adaptations of the initial system designs, is ongoing and will be described in the respective upcoming Deliverables. Some of the related deliverables go into more detail on used data models or the type of data being used internally or being exchanged. In some cases, well-known standards (cf. Annex F) are used, but do not directly relate to the APIs being built, or are sub-aspects of APIs being built and thus do not show up. This also leads to a lower number of APIs directly referencing open standards, together with the needs of partners to build internal prototypes for data exchanges or as part of confidential deliverables. Therefore the following points to some of these related reports.

The Seamless eMobility report (D2.5: Seamless eMobility system including user interface (Skoglund et al., 2020)) describes the work done towards Intelligent Transport Systems (ITS), it includes a brief description of the use of the Transmodel (EU CEN TC278) family of standards (cf. Annex F) with data exchange formats like NeTEx[8] and SIRI (Standard Interface for Real-Time Information) and describes the EU Regulation 2017/1926 (ITS Regulation)[9] and the mandated structure and format of multimodal travel information, both required static and optional dynamic. It further describes the 4C TTC system backend, including data collected and integrated, APIs used, and examples of data used in the project. Details of the eMaaS case are further described as an example in Section 5.2.

The Toolbox of systems for PEB design (D2.2: Toolbox for design of PEB including e-mobility and distributed energy resources (Dahlen et al., 2020)) is a confidential report. It describes 3 tools for PEB design: a bottom-up building modeling tool, a top-down energy modeling tool, and a community grid optimisation model. It describes the used data input, for example time series of energy data, building modeling, grid topology, connection to grid planning tools and backend data sources.

This Community Grid Framework (D2.6: Framework for Community Grid Implementation (Antolić et al., 2021)) describes the organisational setup and also the necessary ICT and energy systems and their deployment. It includes the LoRa/LoRaWAN interfaces within the setup and local installations and messaging/communications to connect to SLUs (Smart Link Units, specific Smart Meter infrastructure), describes the connection of the

[8] http://netex-cen.eu
[9] https://eur-lex.europa.eu/eli/reg_del/2017/1926/oj

enerXchange trading platform with the SLUs and the execution of trades, including the IOTA verification of exchanged data.

The PEB Trading Market system (D2.7: Local DPEB trading market demonstration tool (Livik et al., 2021)) describes the integration of energy assets into the Algotrader platform for trading and invoicing, where assets (and building systems) are linked with the ABB OPTIMAX® Energy Management for Sites (SiteEMS) system through smart meter data. It describes the bid structure for trades and the IOTA verification (see also Section 6), the use of time series databases with FastAPI APIs, the integration of external open data sources such as geographical info, weather, the links to the eMaaS system, and the use of grid topology and asset locations based on grid operator data, for example as GIS or CIM data. It contains detailed fact sheets in its annex.

The Modeling Platform and Decision Support Tool (D1.4: Demonstration of the +CityxChange Integrated Modeling Platform (Purshouse et al., 2021)) is described in a confidential deliverable, while its application is public (e.g. D4.1: Limerick DST (Integrated Modelling and Decision Support Tool) including training manuals/videos (Kerrigan et al., 2020)). It includes an overview of the different types of internal and open data that the tool needs or accepts, for example maps and building data as GIS (Geographical Information Systems) and BIM (building information models) files, energy data in various formats, socio-demographic data. Many of these are exchanged not programmatically, but manually, meaning they are not directly part of the API work described in this report.

Out of the high-level named data standards discussed in Annex F, some are directly used by partners. For example, Transmodel and related ones for the eMaaS (by 4C) FIWARE for automatic KPI exchange (by FAC), Dublin Core as metadata in publication repositories (UL/NTNU), and a number of other ones that are in many cases handled internally and have preliminarily been described in the list above. In other cases, data exchange on the building suite already follows built-in standards of the used solutions and are therefore also of lesser concern to partners on the development. For other development, partners seem to prefer in-house data models based on requirements specification of their system and consultation of existing data models and standards, resulting in a partial and liberal use of the data standard, customised for their system/task.

Further standards are listed in Annex F as an overview and for further reading, as many are not or only indirectly used in the demos and APIs. Further ones will be discussed in other forthcoming Deliverables, including the reports on the adaptation of the built systems and their deployment in specific city demonstrations.

### 4.4.1 FIWARE Data Model use (example from M&E for KPI exchange)

As an example, we show the data model developed for the KPI tracking and monitoring. In this case, FIWARE was chosen as a data model to be adopted/adapted. Implementation of FIWARE standards can help with achieving data interoperability and data integration. FIWARE puts forward a standard data model as well. This example uses the FIWARE-based Data Model Elements and Labels to refine a data model for KPI exchange.
The Monitoring & Evaluation MERT reporting tool for the +CityxChange KPIs is developed and described in D7.4 (Rood, 2019). We reviewed MERT API no. 3 as an example in the API Catalogue together with FAC, and we checked its compliance to FIWARE. This API is concerned with KPI information relevant to 11 Demonstrator Projects in the +CityxChange. FIWARE Data Model has a list of metrics which can be compared to FAC KPIs and we see similarity in the definitions but not full compliance. FAC argues that full compliance is not needed and customisation for +CityxChange is required, with which we agree at this stage. There is again an Agile interoperability management lesson here: we may start with non-full compliance and adhere fully if the next steps dictate it (full compliance is justified when other partners also decide to be fully FIWARE-compliant, it is often not justified to be done by a partner in isolation). In future, in order to increase interoperability and usability across a wider community, we advise that more close FIWARE use/compliance including use of similar data element labels is considered by the partners.

Appendix E contains the definition of the KPI API (from FAC) in the Blueprint standard, which shows their suggested data model (considering FIWARE, but not using it fully) as well. Partners usually prefer solutions exactly matching the task at hand and do not usually fully customise an existing solution in a way it is unchanged. FIWARE Data Model snapshots for KPIs[10], including the labels, follows for quick reference of the partners in Appendix E.

10 https://fiware-datamodels.readthedocs.io/en/latest/KeyPerformanceIndicator/doc/spec/index.html

# 5 API Catalogue and API use cases

The use of APIs in the EA use cases for integration and and their description is a main result of this Deliverable. These results are part of the practical work that has been done between the partners and are one specific explored way to ensure Data Integration and Interoperability. The work of this Deliverable aims to support the API discussion and development between partners with the lightweight framework presented in Section 3 and 4. The technologies and development methods for the respective backends is out of scope of this report and will be handled by partners internally and partly described within their Deliverables. The Development of the API catalogue system is described in the Annex.

As part of the overall ICT Ecosystem, this section describes the API use cases of the API catalogue as they relate to the +CityxChange EA use cases of D1.2. The catalogue is described in further detail in Annex A and the cases separated out in Annex B.

## 5.1 Integration of APIs in the +CityxChange Enterprise Architecture Use Cases

Together with the partners named in the task definition, and with help from other relevant partners, we have developed API definitions and EA Use Cases including API Catalogue references (using the corresponding API numbers in the catalogue, the use case *template* appears in *D1.2: Report on the architecture for the ICT ecosystem* (Petersen et al., 2021)).
We have argued that development of different layers of use cases would benefit partners and consultants/experts with understanding and replication of the results:
Following is the illustration of +CityxChange EA described in D1.2 (Petersen et al., 2021) which is the overarching basis, from architectural point of view, for this present report which provides a framework for the interoperability and DataxChange (Data Space). It is also the basis for documenting the use cases in which the APIs are also referred to with their API numbers in the API Catalogue. The overall ICT ecosystem is shown in the figure below with the services layer highlighted in red as seen which captures services such as the eMaaS, etc.

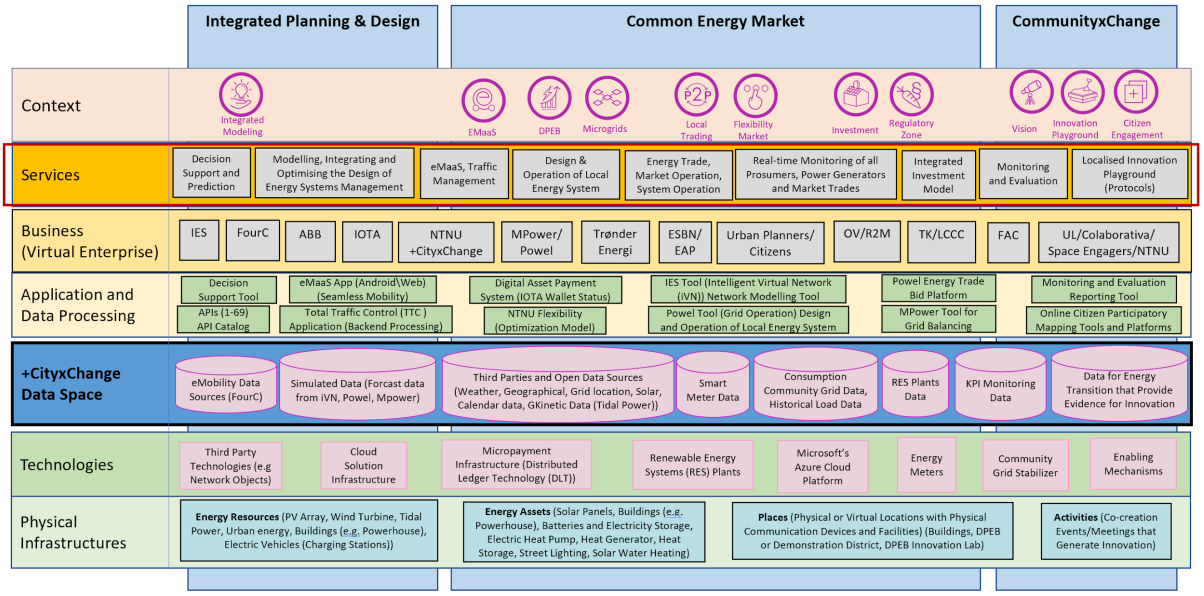


Figure: Overall ICT Ecosystem for the +CityxChange project, with the icons in the context layer representing some of the use cases (source: D1.2 (Petersen et al., 2021))

Additionally, API Blueprint was advised to partners for defining the APIs while the information provided by partners is incrementally added to the catalogue (as provided by the partners) as they still continue developing APIs (as an ongoing process, so this Deliverable contains the state at time of writing).
By completing the API Catalogue with partner API information, and in some cases with the suggested API design, for a partner who has not yet developed the APIs, and by documenting the data exchange diagrams relevant to the partner case, we will have an API/Interoperability case which should be shown under the EA case as a whole for the partner and be accompanied by some details and explanations. By doing this, in a bilateral conceptualisation-feedback loop, we show how the partners' system and technical/business collaborations can be modelled and possibly enhanced using +CityxChange EA and we receive feedback from the partner on the EA, specially (in the scope of this D1.3) on the APIs and Interoperability part (relevant to the DataxChange/Interoperability part of the +CityxChange EA).

In the above ICT Ecosystem, the DataxChange layer may include APIs which facilitate exchange of data, making that layer support the data-as-commodity principle, as the data is not usually exchanged among firms by direct access to their databases, but it is exchanged through structured APIs. On the other hand we may move APIs to the Application and Data Processing layer and call the underlying layer Data Space (containing DBs), supporting the fact that APIs are application elements. The overall ICT Ecosystem, the final EAF, the Use Case Template and the EA use cases appear in D1.2 (Petersen et al., 2021).

As the cases are discussed in D1.2 (Petersen et al., 2021), the cases or their analysis are not the main contribution in this report, but rather we provide examples (specially to show API use in the cases). This section describes the APIs used in the +CityxChange project. The APIs employed in the eMaaS case (presented in D1.2, Section A.1) are described in the following Section 5.2. The APIs employed by partners for the remaining use cases (17 overall, cf. D1.2) are covered in Appendix B.

## 5.2 API Example: Seamless eMobility System Including User Interface

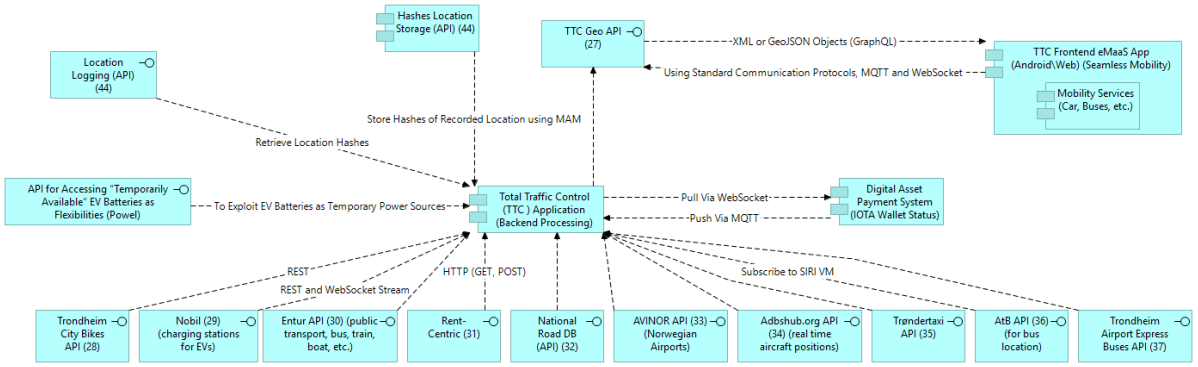


*Figure: Application and data processing layer for eMobility system (source: D1.2)*

Here we describe one of the EA use cases in detail with respect to the API development and use, namely the eMaaS system, mostly with the data integration view.. As seen in the above figure, the application and data processing layer comprise different APIs (as seen in the following table) that provide data to the Total Traffic Control (TTC) application (backend processing) developed by 4C in Trondheim, Norway. Also, the protocols and standards (such as MQTT, XML, REST, websocket stream, GeoJSON, etc.) employed by the APIs to connect a digital payment system (by IOTA) are captured. The full use case for the seamless eMobility system including user interface is presented in D1.2. A description of the APIs used in the use case (based on the graphical view in the figure above) is shown in the following table.

| API number | API name | API owner | API consumer | API description |
|---|---|---|---|---|
| 27 | TTC Geo API | FourC | eMaaS application | Uses REST APIs to provide information from TTC application. |
| 28 | Trondheim City Bikes | City bikes | TTC application | Uses REST APIs to provide information about the stations where the bikes are parked. |

| | | | | |
|---|---|---|---|---|
| 29 | Nobil (charging stations for EVs) | Nobobil | TTC application | Uses REST APIs to provide a national registry of charging stations for EVs. |
| 30 | Entur (public transport, bus, train, boat, etc.) | Entur data | TTC application | Provides access to the national registry for all things involving public transport, bus, train, boat, etc. |
| 31 | Rent-Centric | Car sharing data (AVIS) | TTC application | Mobility solution provider that offers information about parking and EVs location. (currently not used due to change in providers) |
| 32 | National Road Database | Road datex | TTC application | Provides roadside signs and taxi ranking. |
| 33 | AVINOR (Norwegian Airports) | Flight info | TTC application | Handles and owns most Norwegian airports and provides flight information. |
| 34 | Adbshub.org (real time aircraft positions) | Third Party Data | TTC application | Provides a community driven service that provides real time aircraft positions. |
| 35 | Trøndertaxi | | TTC application | Provides information on taxi positions. |
| 36 | AtB (for bus location) | AtB Bus DB | TTC application | Provides information on good, updated, accurate and correct bus positions. |
| 37 | Trondheim Airport Express Buses | Third Party Data | TTC application | Provides information on real time positions of Værnes-ekspressen airport express buses. |
| 44 | Location Logging | IOTA | TTC application | Provides data that logs the location of the traveler. |
| 44 | Hashes Location Storage | IOTA | TTC application | Stores the location data of the traveler. |
| # | "Temporarily Available" EV | Powel | TTC application | Provides data that simulates temporarily available EV |

| | | | | |
|---|---|---|---|---|
| | Batteries as assets | | | batteries as energy flexibility assets. |

*Table: Description of the APIs in the eMaaS use case*

The table above for this example has a number of API types, which we explain here.

The following description of the APIs for this case is taken partly from the respective Deliverable D2.5: Seamless eMobility system including user interface (Skoglund et al., 2020).

Third Party Data Provider APIs: A summary of various interfaces and APIs that are used as data providers to add mobility information into the TTC eMaaS backend. Input sources are at this state mainly from Trondheim.

- Trondheim City Bikes (API no. 28): firm which provides bikes for rent at 55 locations around Trondheim, allowing ease of travel around the city. Information about the City Bikes in Trondheim are available through two REST APIs. The first provides information about the stations, and the other provides real-time data about how the stations are operating.
- Nobil (API no. 29): a Norwegian national registry of charging stations for electric vehicles, which allows electric car owners to easily locate the nearest charging station. Nobil Delivers this data through two methods, a REST API and a WebSocket Stream.
- Entur (API no. 30): Entur is a national registry for all things involving public transport; bus, train, boat, etc. They provide static information about stops and routes, as well as real-time information about vehicle locations, delays etc. The quality of information depends on the source data from the operator that provides it to Entur.
- Rent-Centric (API no. 31): a Canada based mobility solution provider for possible AVIS initiatives. Their APIs provide information about parking locations and the vehicles there. The API is not publicly available. Currently not used due to change in providers.
- National Road Database (NVDB) (API no. 32): for taxi ranks out of a variety of data.
- AVINOR (API no. 33): AVINOR handles and owns most Norwegian Airports. They offer an API which provides information about incoming and outgoing flights.
- Adbshub.org (API no. 34): a community driven service that provides real-time aircraft positions. In order to get data, you need to contribute by providing air traffic data, which FourC currently does. FourC will use this API to match flight info from AVINOR to real-time aircraft positions. This mapping is not obvious, and a specifically crafted interface for this is being made.
- Trøndertaxi (API no. 35): Major taxi service in Trondheim.

- AtB public transport authority (API no. 36): In order to have good, updated, accurate and correct bus positions, we need to subscribe to SIRI VM from AtB's own services, and not Entur, as it is currently the case.
- Trondheim Airport Express Buses (Værnesekspressen) (API no. 37): Real-time positions of Værnesekspressen airport express buses. FourC hosts the fleet management solution for this bus service and is providing a SIRI VM interface.
- IOTA APIs (APIs no. 8 to no. 25): recording and retrieving journey and payment information using IOTA's DLT called Tangle

Service APIs: These are the APIs that the 4C TTC system offers

- The TTC API (API no. 73)
    - SIRI VM interface for vehicle position. Currently, only PT vehicles are available. Other modes of transport can be added when standardized in a newer SIRI/transmodel version.
    - NeTEx - route network data. Currently, supported for PT routes only.
    - Bus stop position - TTC offers an export of bus stops, using concepts like Quay and StopPlace, as defined by Transmodel. Currently supports all stops defined in NVDB.
    - Charger data - Positions and status/attributes, historical and real-time
    - EV data - Core data like positions and status/attributes, historical and real-time
    - Vehicle availability
    - Journey description
    - Road closures and status
    - All data are stored historically. This means that the current status of an object and its attributes for a specific moment back in time can also be queried for.
- TTC Geo API - eMaaS API (API no. 27)

  "The TTC Geo API is designed [...] for integration with different map solutions. The API is mainly designed to output geographical data, which can be displayed and interacted with on maps. The objects themselves are data rich, but it's up to each individual user of the API to utilize the information that is made available. [...] This API delivers data as GeoJSON objects (RFC 7946). [...]The API is based on GraphQL, a modern alternative to the more traditional REST." (Skoglund et al., 2020)

This use case also includes specific Transmodel standards: "The TTC backend is built on EU CEN TC278 standards for public transport [...], popularly named Transmodel. Transmodel is a family of standards, in which also data exchange formats like NeTEx and SIRI (Standard Interface for Real-Time Information) are included.[...] Traditionally, Transmodel specifies a

model for use with ordinary public transport using traditional transport modes like buses, trams, trains, etc. CEN TC278 standards, however, are being extended to include new entities in newer versions of the model. FourC keeps its implementation of data types, formats, methods and architecture aligned with TC278's current and planned representation of these new objects. Likewise, FourC will align the implementation with possible plans on national levels for these issues." (Skoglund et al., 2020)

## 5.3 API Example: IOTA eMaaS Payments Trail and Tech Specs and Requirements

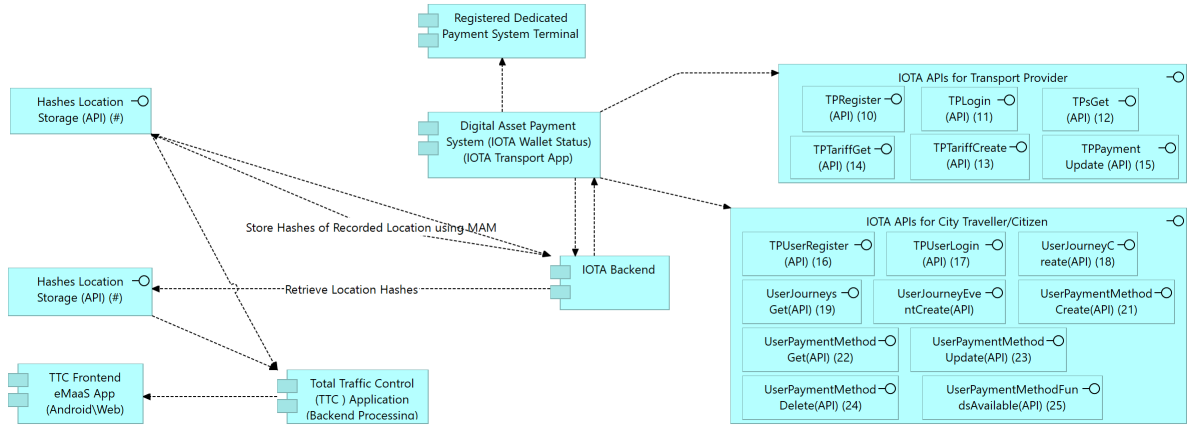


*Figure: Application and data processing layer for IOTA eMaaS Payments Trail and Tech Specs and Requirements (source: D1.2)*

As a second related use case, we describe here the case of the potential IOTA payment integration into the eMaaS system, which is ongoing at the time of writing. Based on the use case for the IOTA eMaaS Payments Trail and Tech Specs and Requirements use case as discussed in D1.2 (section A.2. IOTA eMaaS Payments Trail and Tech Specs and Requirements Use case). Further details can be found in the respective D2.5 (Skoglund et al., 2020) and in Section 6. The APIs deployed for that use case are shown in the following table.

| API number in catalogue | API name | API owner | API consumer | API description |
|---|---|---|---|---|
| IOTA APIs for Transport Provider APIs (10, 11, 12, 13, 14, and 15) | TPRegister (API) (10), TPLogin (API) (11), TPsGet (API) (12), TPTariffCreate (API) (13), TPTariffGet (API) (14), and TPPayment Update (API) (15) | IOTA | FourC | Provides terminal that processes booking and payment made from the traveler/citizen to the respective TP. |

| | | | | |
|---|---|---|---|---|
| IOTA APIs for City Traveller/Citizen (16, 17, 18, 19, 20, 21, 22, 23, 24, 25) | TPUserRegister (API) (16), TPUserLogin (API) (17), UserJourneyCreate(API) (18), UserJourneysGet(API) (19), UserJourneyEventCreate(API) (20), UserPaymentMethodCreate(API) (21), UserPaymentMethodGet(API) (22), UserPaymentMethodUpdate(API) (23), UserPaymentMethodDelete(API) (24), and UserPaymentMethodFundsAvailable(API) (25) | IOTA | FourC | Provides data for eMobility and offers a terminal that processes booking made from the traveler/citizen to the respective Transport Provider (TP). |
| Hashes Location Storage API (44 and 45) | Hashes Location Storage | IOTA | FourC | It processes location logging and hashes location storage of the traveler/citizen via API 44 and 45. |

*Table: Description of the APIs in the IOTA eMaaS Payments Trail and Tech Specs and Requirements use case*

These APIs, based on the D2.5 case, would allow a user to book and record a travel on the IOTA Tangle, thus providing an audit trail and guarantee integrity of required payments to be distributed to a transport provider, while these APIs allow to process micropayments for each trip from user to transport aggregator to transport provider.

The APIs used for the eMaaS case as shown in the above table and comprise the getVersion API (API no. 8) which is used to access API details such as version, while getDocs (API no. 9) is used to access more details, i.e. the documentation for the APIs.

Furthermore, the transportProviderRegister API (no. 10) is used to Register a new transport provider in the system, while transportProviderTariffCreate API (no. 13) can create the tariffs for the transport provider. The transportProvidersGet and transportProviderTariffGet (APIs no. 12 and 14) are used to get a list of transport providers and their tariffs respectively. Getting the tariffs should be done one by one, one time for each provider queried. The transportProviderLogin (API no. 11) is used to log a previously registered user into the system. The transportProviderPaymentUpdate (API no. 15) updates the payment details for the transport provider, such that the transportation service provider can receive payment for the service. Using the userRegister and userLogin APIs (no. 16 and no. 17), a new user can be introduced to the system (once) and then login to the system as needed afterwards (these are of course done through a User Interface and the APIs are in fact

called by the software which provides eMaaS). API no. 18, userJourneyCreate, facilitates creation of journeys for the registered user. Summary list of all the user's journeys can be retrieved using userJourneysGet (API no. 19). An event can be added to a journey using API no. 20, userJourneyEventCreate. The APIs userPaymentMethodCreate, userPaymentMethodGet, userPaymentMethodUpdate and userPaymentMethodDelete (APIs no. 21 to no. 24 respectively) are used to create, know about, change or delete the payment method using which the eMaaS service user pays the service provider. The userPaymentMethodGet API provides a summary for all the users while the other three APIs are used user-by-user. Finally API no. 25, userPaymentMethodFundsAvailable, can be used to check whether a user has enough funds available in their payment method. IOTA's Tangle, is used to store and access journey and payment data in the above APIs, making them a good case showing application of DLTs for Smart City Transportation services. Most of the mentioned RESTful APIs use JWT tokens for authentication (except the register APIs, getVersion and getDocs), and communicate via JSON (except getDocs, which uses XML/HTML).

## 5.4 Brief analysis of API catalogue and API structures

We have encouraged partners through D1.3, in Data Meetings and the consortium as a whole to use and build open APIs and report them in the API catalogue.

The actual development of APIs has been done partly in this task and mainly in other respective tasks that build and adapt the systems, in close connection to the other work packages.

At the time of writing it contains 69 APIs (and two deprecated ones). Additional ones are still being developed or adapted as part of the local deployments. The API catalogue is presented as a table in Annex A (extracted key information, with confidential information removed). In the following we provide a brief analysis of its content.

REST is the preferred protocol among partners (68 overall), as is now common in web applications. No one reported the use of Websockets. The remaining one is GraphQL as an advanced protocol offering higher flexibility and efficiency than REST.

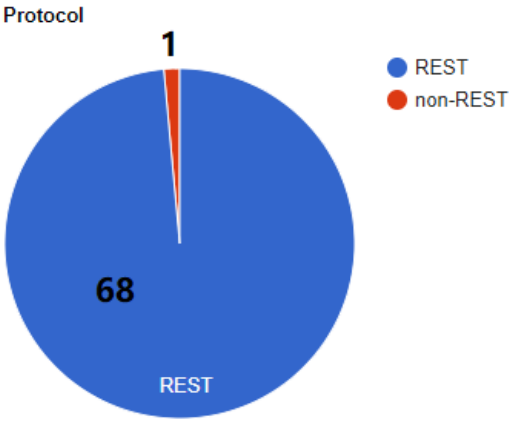


*Figure: Protocols as reported for the APIs in the Catalog (mainly REST, one GraphQL and WebSockets)*

JSON is the preferred *data format* standard for the APIs in the consortium (67 entries), while XML use is limited. This is in line with the use of REST, which in most technology stacks transmits JSON data.

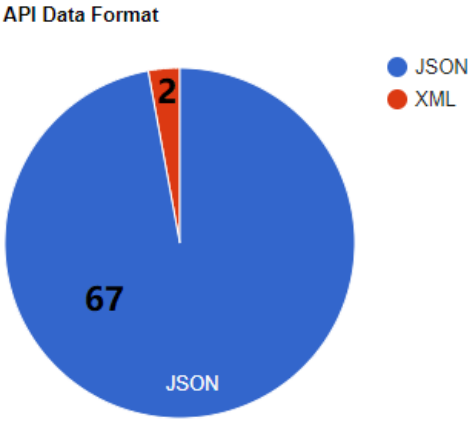


*Figure: API Data Transport Formats as reported in the API Catalog*

Of 69 APIs gathered, 61 are reported as publicly accessible (some have been under development, and are intended to be opened later as reported by the partner).

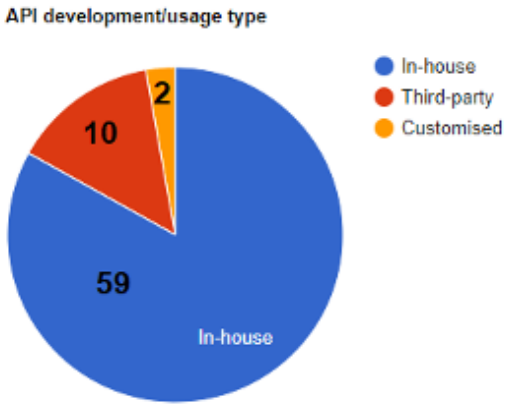


*Figure: API development and usage type (in-house vs. external)*

In terms of make-or-buy, 59 APIs have been built in-house, 10 come from third parties, and 2 are customised 3rd party APIs.

We briefly assess quality of the provided metadata in the catalog with two simple metrics:

- Description availability: whether there is a description that gives an understanding of the API scope. This is mostly fulfilled at least with basic description; out of the 69 APIs, 68 have a description with 1 missing.
- Specification availability: whether there is a specification (or documentation, or link to specification) that allows to understand the API use. Most APIs have the direct specification added, and some have additional code examples, detailed documentation, or links to further specifications. Out of the 69 APIs, 65 have some form of a specification and 4 do not have it.

For the use of data standards within the APIs, the ones mentioned in the API catalogue are extracted in Annex A, though these mostly concern data transport/exchange. A brief discussion of data models is given in Section 4.

Most of the contributions into the catalogue have been done during or in connection to data meetings or one-to-one meetings (estimated around 80% of entries), while content prepared outside of the meetings (and APIs developed in multiple other tasks and work packages) has been contributed. This shows the important role of these facilitation mechanisms in getting and completing the information and supporting interaction. Especially the followup and one-to-one meetings were important to complete entries. Much of the content was specifically prepared, but in several cases it could be generated from internal technical documentation or specifications, access guidelines for open data portals, or ongoing development and documentation activities.

This emphasises a lesson and experiences in the interactions: the practical and direct work with the partners is very important. Much activity also happens outside these structured artifacts and in areas where agreement is only needed between two partners for their direct integration. While this is in line with the distributed overall ICT approach and the separation and delegation of concerns, it leaves some developments invisible to the present work. Ensuring data integration and interoperability in a consortium is itself a complex task with multiple sub-projects. A different structure of this task or a different anchoring of interoperability activities either within WPs or at the PM level may help address these issues. However, through some structures such as the need to deliver M&E data, and the involvement of the cities as actors within the data space, it could be ensured that cities as project owners are aware of these issues and they could ensure that their main data and data exchange needs were considered within these tasks.

## 5.5 Additional guidance: Selected best practices

As part of the interactions between the partners and the discussions around architecture and API integration, a number of guidelines were reviewed and discussed. These range from very general initial guidelines of general Web API development up to very specific ones on enterprise integration and maturity levels. A vast array of literature for specific development or management is also available, for example on API management (Medjaoui et al., 2018) or Web APIs (Amundsen, 2020).

We briefly note wider guidelines here as part of the DIIF implementation support and higher levels of the framework. There are many more available for specific use cases, specific domains, or collections of specific examples and other best practices.[11]
In particular:

- The next section on *Implementing APIs* describes overall implementation guides for APIs. This is quite an extensive list of general and specific best practices. The full list is found for reference in *Appendix I – API Implementation Guides*.
- The following section describes 10 steps of overall best practices in *API Documentation* to ensure easier access and reuse between organisations
- *Appendix G – Related European Guidelines & Tools*: describes relevant European standards and guides for API implementation within the overall European Interoperability Framework EIF, which is also the basis for this document
- *Appendix H – Implementation Advice for Partners on EIF*: This is an extension of the EIF guidance, with specific implementation advice developed and added based on partners and project needs. It helps to make the EIF more actionable and provides additional structured support.

### 5.5.1 How to Implement Good APIs: API Technical and Data standards

Web based APIs need to consider and implement a number of best practices and benefit from the combined experience of the community which has been put forward in the form of rules and reminders, such as the over 100 sub-rules extracted from the 33 rules/rule-categories suggested by API Technical and Data standard (v2, 2019, British Government[12]). We advise partners to use these for their API implementations. In Appendix F, we mention the 33 rule categories and also mention the 100+ sub-rules extracted from these 33, for quick and easy reference of partners (although we do not aim to mention all the possible sub-rules). Please refer to **Appendix I** for the important rules which we advise partners to use for API implementation.

[11] we refer to just one technical example here out of the wide ranges available. https://dret.github.io/guidelines/
[12] https://www.gov.uk/guidance/gds-api-technical-and-data-standards

### 5.5.2 API Documentation Best Practices

It is important to document APIs well such that others are encouraged to use the APIs and are helped when trying to use the APIs (specially open APIs). A number of Best Practices for documentation of APIs have been considered. The simple white paper[13] we advise partners to benefit from (through 5.8), titled *"The Definitive Guide to Creating API Documentation"*, covers ten best practices:

> "1. Follow the standard template or outline for organizing reference pages.
> 2. Use a terse, factual writing style. Sentence fragments are desirable. Avoid adjectives and adverbs.
> 3. Provide complete information about each API component.
> 4. List all error messages alphabetically, specify the level of the message, and provide suggested workarounds and solutions.
> 5. Provide working code snippets for each method, function, and resource. You don't need complete examples, but show a common use of that element.
> 6. Provide flow charts showing the sequence of the most commonly used methods for common use cases.
> 7. Provide sample programs demonstrating common use cases.
> 8. Provide a "Getting Started" guide showing how to develop a program for common use cases.
> 9. Provide performance and tuning information.
> 10. Provide a contact in case developers have questions or need additional assistance."

[13] https://assets.madcapsoftware.com/white-papers/White_Paper-The_Definitive_Guide_to_Creating_API_Documentation.pdf

# 6 Potential of DLTs as Data Exchange Mechanism

In +CityxChange, DLTs (Distributed Ledger Technologies) are explored as a potential technology to exchange immutable data, validate data integrity, or perform micropayments within the peer to peer energy trading (D1.2 section A.10) and eMaas eco-systems (D1.2 section A.2). Here we provide a summary of the IOTA involvement so far, and give an initial assessment of the use for the interoperability and data exchange challenges.

The main project aim is the development of Positive Energy Blocks (PEB), portions of a city which generate more energy than they consume and can support local energy trading, also linked with integrated e-mobility (see D2.5 (Skoglund et al., 2020)). This requires to exchange energy excess and energy flexibility, to balance energy use and need between buildings, prosumers, and energy assets (electricity use, heating, smart meters, batteries, solar panels, heat pumps, electric vehicles, chargers, V2G, etc.) (see D2.7 (Livik et al., 2021)). It is a software-heavy undertaking, which requires examining ways to connect building energy systems, enable trading, exchange data, and provide micro-payment solutions in this novel ecosystem (D1.2 section A.10 and section A.2).

A large number of small peer-to-peer transactions are expected within the local grid, which mandates a low or zero-fee micro-payment protocol. Existing PoCs towards data markets are expected to provide value and insight. For an energy and sustainability project, it is important that the energy demand of the solution does not eat away potential energy savings. Therefore, we believe that any projects, and energy projects in particular, should use energy-efficient systems towards Green Computing.

Since DLTs are a relatively new technology, fallback options are going to be used in many cases in the developed demonstrations. Details are given in the individual demo cases below.

## 6.1 Background on DLT and IOTA

DLTs are represented by the +CityxChange partner IOTA Foundation, and IOTA integration or prototyping of solutions was planned in the main energy trading and mobility systems (D2.5 Seamless eMobility system including user interface (Skoglund et al., 2020)) (D2.6 Framework for Community Grid Implementation (Antolić et al., 2021)) (D2.7: Local DPEB trading market demonstration tool (Livik et al., 2021)) to gather and integrate experience with DLTs. DLTs are a way for distributed nodes in a network to exchange data and reach consensus on their validity without need for a central authority and database (as a trusted data exchange infrastructure). Many DLT/blockchain initiatives have been started in various sectors such as energy, transportation, economy, healthcare. Most domain specific

applications are still in a prototype stage. We refer here to an overview of urban and smart cities applications (Shen & Pena-Mora, 2018).

DLTs are only applying their whole strength in specific scenarios. Requirements such as noted in the previous section and features such as data contracts, cryptographically secured manipulation-free operation, distributed operation, anonymity, etc. may individually be solved with other existing technologies, but in combination may need DLT. Requirements analysis includes the distinction between private or public access, need for anonymity, need for control, etc. In application design, the use of cryptocurrency can be second to these other questions. Multiple decision models exist to understand whether DLT is applicable and suitable for a specific application.[14]

The arguably most well-known blockchain systems are BitCoin and Ethereum. These are usually not very energy-efficient as they demand a proof-of-work to verify transactions through a consensus protocol. A lot of computing power is needed to operate the network and super nodes, called miners, with expensive hardware being required. In addition, transaction fees are required, which are in turn paid to the miners for successful operation as an incentive to operate the network. In +CityxChange, there was the need to explore more energy-efficient DLTs solutions.

IOTA has developed a different DLT approach based on DAG (directed acyclic graph), which represents the "Next Generation Blockchain" (see IOTA Research papers[15]). IOTA was designed specifically for the Internet of Things (IoT). Its consensus protocol requires less computation overhead and has no fees. For micropayments in IoT and between devices and minimal amounts of energy traded, any fees larger than the amount of value transferred would be infeasible. In this way the IOTA Tangle is designed to enable permissionless, tamperproof, fee-less and real time peer-to-peer IoT data and value transfer on an open source platform, between any actors (machines, humans, organisation etc.).[16] IOTA is also established as one of the lowest energy consuming DLT networks.[17]

By introducing the above innovations, IOTA fulfils the following requirements for a DLT for energy in smart cities (cf. IOTA smart cities whitepaper (Pimenta de Miranda, 2019)): permissionless network; zero fee structure; Open Source; high transaction throughput in line with expected number of transactions; low energy footprint.

[14] Meunier S., When do you need blockchain? Decision models., medium.com, URL: https://medium.com/@sbmeunier/when-do-you-need-blockchain-decision-models-a5c40e7c9ba1
[15] https://www.iota.org/foundation/research-papers
[16] https://blockchain.ieee.org/technicalbriefs/january-2019/iota-feeless-and-free
[17] https://greeniota.com/docs/

## 6.2 IOTA Tangle in +CityxChange

The IOTA DLT supports use cases of trading within the PEBs and between assets within them, through concepts for fully integrated solutions and implementation of proof of concepts (POCs). Here are the +CityxChange use cases that IOTA technology can support: automated trading systems, pricing and markets, contracting and billing, e-mobility integration, and tracking and validation of (green) energy use.

In the current design, and since DLTs are a relatively new concept in this domain, fall-back mechanisms are being set up. In some cases, these are also mandated to comply with regulations in the energy sector, which bring additional non-technical requirements into the analysis. The project also works with regulatory sandboxes and mechanisms, for example a sandbox for P2P energy trading (+CityxChange Deliverable D2.1 (Bertelsen et al., 2019)). This is a concept derived also from the EU financial sandboxing.[18] IOTA was keen with the other project partners to enable micropayments in the areas of eMaaS and energy trading; however, due to regulation uncertainty and technical considerations there was the preference to not implement any "real" payments within their use cases. For this reason, use cases involving cryptocurrency payments have been developed to the stage of prototypical PoC using a private network where no monetary value is directly transferred.

For the fall-back mechanisms and alternatives, to ensure unhindered system operation, one example is to have separate data exchanges using centralized databases and in parallel sharing immutable data hashes through IOTA to ensure data integrity. Another fall-back mechanism was to use traditional billing methods instead of implementing cryptocurrency payments. This also allows partners to test out solutions and switch out the more experimental components, without changes to the core data architecture.

In some cases, a full Tangle implementation in the complete technology stack is not yet possible. For example having IOTA code running within cars or smart meters in the wide deployment would currently not be feasible with legacy off-the-shelf hardware. For the same reason, integration into billing systems may currently not be feasible. In these cases, IOTA is running in parallel to some core backend components of the data transfer with the options to extend this along the axes of value chain and of system integration using IOTA-enabled hardware.
In this case for certain prototypes, an integration of IOTA wallet code into end devices is being prototyped, such as specifically adapted chargers and integration into a car's

[18] European Commission (2018). FinTech: Commission takes action for a more competitive and innovative financial market (press release). http://europa.eu/rapid/press-release_IP-18-1403_en.htm

onboard computer in a separate project,[19] or the case of adding a separate IOTA-enabled IoT device directly to a data port of an adapted smart meter as in the example below. Such thin and lightweight prototypes or parallel deployments are being set up to validate the technology and prepare for a potential later full deployment.

In the following, specific prototypes are summarised from their other Deliverables and related activities.

### 6.2.1 POW energy trading platform (integration)

This section is mainly based on the descriptions in Deliverable D2.7: Local DPEB trading market demonstration tool (Livik et al., 2021), and repeats some content from there, while also giving the IOTA perspective.

The local energy partners were not ready to implement a fully decentralized energy marketplace, partly for regulations limitation but also for organisational and commercial reasons. Because of that, it was decided to implement a semi-decentralized marketplace on which required data are shared using central databases but the ledger offers a way to verify data integrity and immutability before a settlement is performed. In the Powel trading market demonstration, IOTA was integrated to provide a data integrity and verification service (D2.7, Section 4.8).

This service was deemed as a benefit to Powels energy market place and was derived from requirements from the main project partners involved within the D2.7 work (Powel, ABB and TE). The service introduces the concepts of data verification and integrity to increase the trust in the data shared among stakeholders and systems, namely, Powels market place, ABBs OPTIMAX® Energy Management for Sites (SiteEMS) and TronderEnergi's billing service via a central database that would require trust in the third party hosting it. This data can now be anchored to immutable fingerprints onto IOTA ledger, while shared using central databases. In the current implementation and integration, the IOTA service runs in parallel to partners' traditional data pipelines (centralised servers), adding the benefit of passive data integrity and active data verification. (*Data Integrity* - The guarantee that data has not been modified/corrupted/tampered with; *Data verification* - The active check that the data you have matches exactly the data that was logged to the tangle).

[19] IOTA blog, Earn As You Drive with Jaguar Land Rover and IOTA, https://blog.iota.org/earn-as-you-drive-with-jaguar-land-rover-and-iota-3c744d8c0cba/; youtube, Jaguar | Earn As You Drive with 'Smart Wallet' Technology, https://www.youtube.com/watch?v=9pzd4MPy1AI

The architecture of the Integrity and verification service is repeated in the following figure. Energy assets or systems exchange data between themselves as part of the larger system. For certain information from the market value chain that should have its integrity assured, the market system can send a copy (or its hash) into the IOTA Tangle through a developed API. This data can then be verified through the use of IOTAs APIs or can be read straight from the Tangle with integrity assured. For any data that is being written, and then read straight to/from the tangle, this also simplifies integrating multiple complex systems together. For example, all actors within the marketplace only have to interact with IOTAs APIs to parse data to each other with guaranteed integrity. Simple, standardised integration across multiple tools. Further details on the high-level architecture are given in the use cases in D1.2 and here in Section 5 and the respective Annex B.

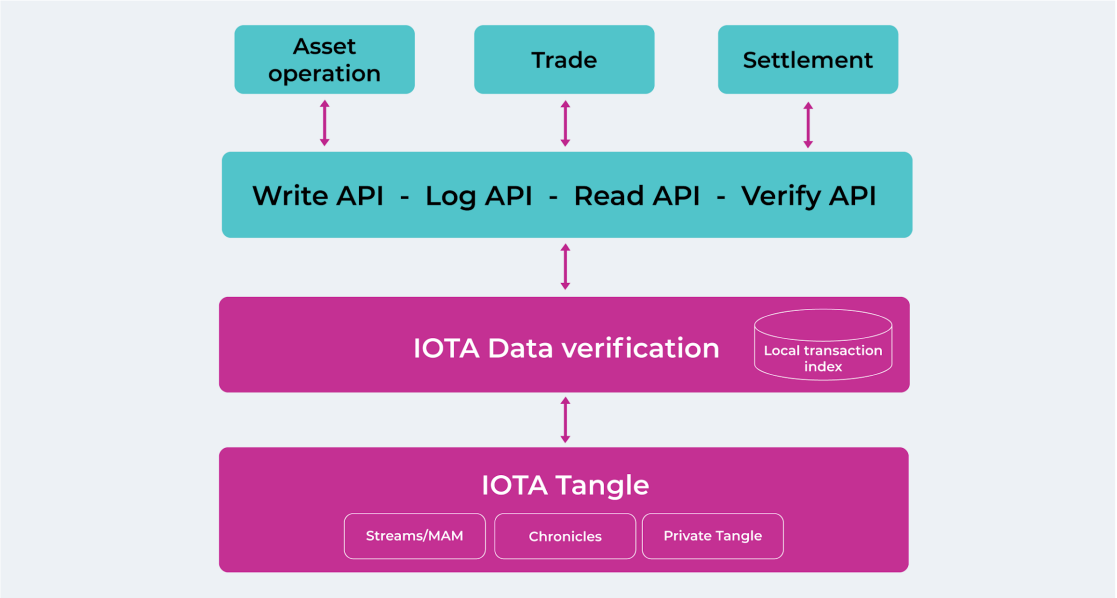


*Figure: Architecture for the IOTA verification service. (Source: D2.7 (Livik et al., 2021))*

It should be technically possible to completely replace traditional data pipelines with the IOTA tangle through the use of hosted IOTA nodes at partner agencies and direct integration of legacy systems. This would be considered the "full" IOTA service and would be considered fully decentralised. This could be implemented with current APIs should partners wish to do so. Implementing this service would be the most secure and beneficial way of utilising the IOTA tangle for data transfer, adding passive data integrity to all data with no need for any active check, be it manual or automatic. Moreover, the scalability of the IOTA ledger would not pose any challenges to the amount of shared data and expected network throughput.

Further information can be found as Documentation (Data logging and verification Technical specification)[20] and Documentation (Data logging and verification integration).[21]

### 6.2.2 IOTA Energy Marketplace and trusted edge energy data (POC)

In order to meet the relevant task descriptions as stipulated by the grant agreement, and to overcome internal project frictions that did not allow a fully IOTA backed marketplace, with full data transfer and IOTA token micropayments, IOTA developed a stand-alone proof of concept (POC) that is in line with T2.3, T2.5, T4.9. (D2.7, Section 5.1, some parts repeated here) for Peer-to-peer payments in a decentralized energy trading marketplace:
Data is captured and validated right at the source, for example smart meters. Then the marketplace conducts bill settlement via information shared on the ledger and IOTA tokens. This concept assumes that each energy asset can have its own IOTA wallet as shown in the following figure. To showcase this, additional software backends and hardware devices have been prototyped (see section below for more details). To fully realise this service it would need to be integrated with energy systems partners and locally deployed systems and devices in a long-term perspective.

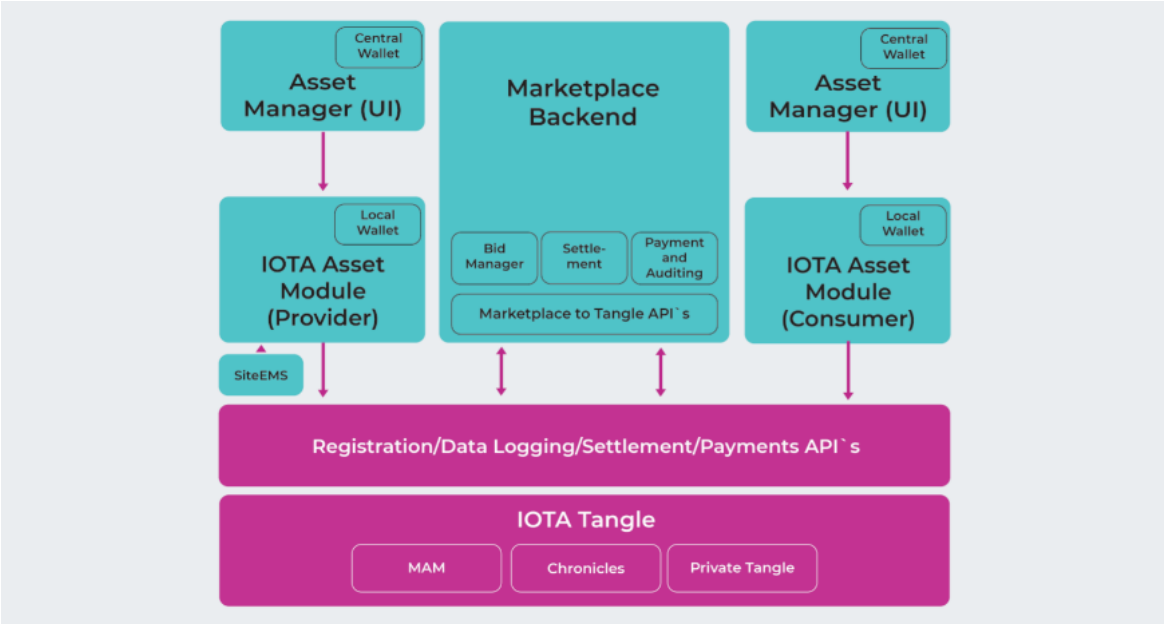


*Figure: IOTA implemented architecture in the prototype energy trading market. (Source: D2.7 (Livik et al., 2021))*

The figure below shows the workflow possible with the provided Marketplace implementation.

[20] https://docs.google.com/document/d/1Asa5ESpuPyhRPiH053UTFdq9JRatMbHXQS7dFtXBu6Q/
[21] https://drive.google.com/file/d/1rtfO953nzJuPsCBZwN85AmkbCw-8ir0y/view?usp=sharing

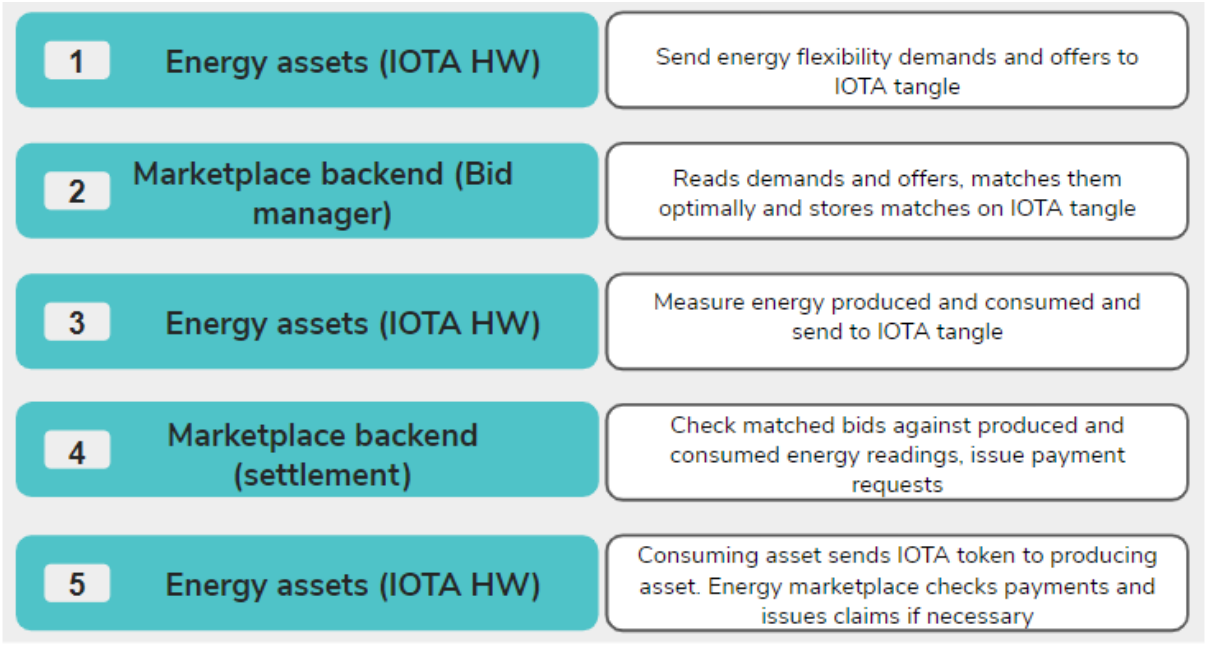


*Figure: Workflow description - Energy marketplace (Source: D2.7 (Livik et al., 2021)).*

The Energy Marketplace interface is shown in the figure below and allows users to configure energy assets to implement specific buying and selling strategies as well as to load assets' wallets in order to have enough token to fulfill the settlements following the marketplace approval of a specific trading.

In particular, using the marketplace front end you have the following functionality:

- Main wallet that stores IOTA tokens per user (add, withdraw from external wallets)
- Wallets that store IOTA tokens per device (multiple per user) (add, withdraw from main wallet)
- Set minimum energy sell price (producer)
- Set maximum energy buy price (consumer)
- Each device has unique ID and advertises through the backend how much energy it wants to sell/buy
- Trades are matched and conducted (currently simulated as no full integration was available)
- Transaction dashboard showing transaction ID, timestamp, energy traded, cost, status

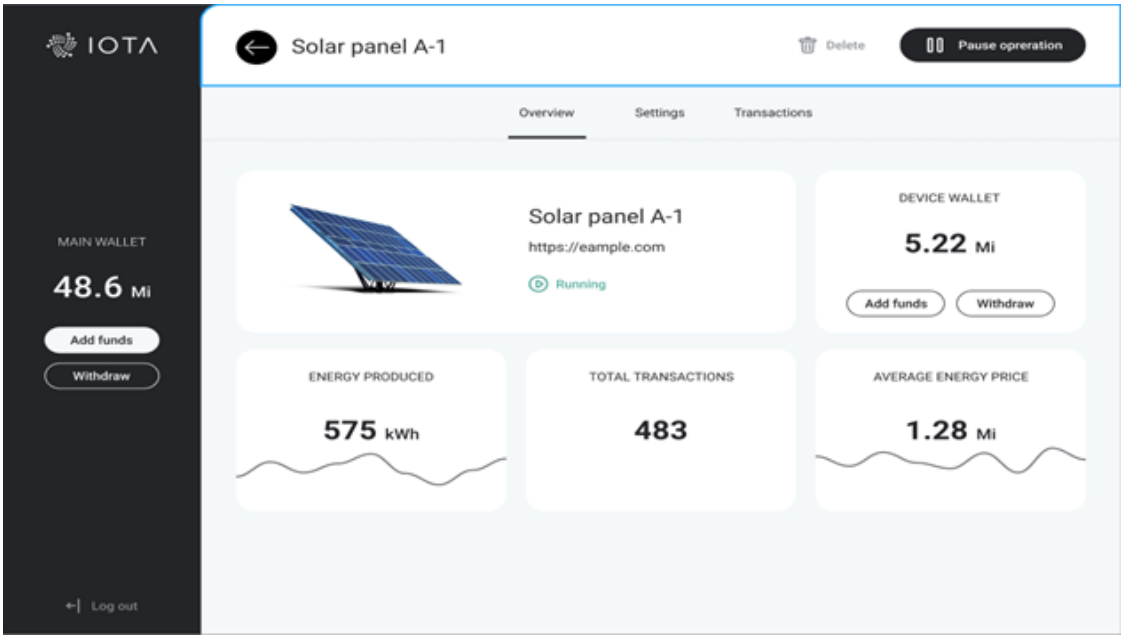


*Figure: Energy market place frontend (Source: IOTA, similar one described in D2.7 (Livik et al., 2021)).*

Using the provided frontend UI, the IOTA energy marketplace is a fully usable, configurable marketplace that can connect to edge HW (producers/consumers), reading energy production and consumption through the developed IOTA device and issuing this data directly to the Tangle. The data can therefore be assumed to be 100% legitimate and untampered with from the device itself.

Signing and verifying data transactions in order to guarantee their integrity and immutability requires the edge devices to be able to be programmed for creating IOTA transactions. This is not possible with legacy devices and requires the use of proxy or virtual devices replica in the cloud. However this does not guarantee full end-to-end security of shared data. For this reason at this stage, before smart meter devices become compatible with ledger functionalities, IOTA faced the need to develop dedicated hardware to demonstrate the whole value chain.

To bring IOTA closer to the smart meter, an asset measurement device has been prototyped. Currently it is not possible to add IOTA code directly to the smart meters for technical and regulatory reasons. Moreover there are many energy devices (such as PV panels, etc) that might not be behind a smart meter. The proposed approach of a dedicated IoT meter with IOTA inside, allow to measure energy directly from not metered devices (i.e., through an energy clamp) or connected to the data port (HAN/modbus) of an existing smart meter or asset manager (i.e., ABB OPTIMAX® SiteEMS).
This is an embedded IoT device (based on a PyCom board) with an IOTA Cryptocore that can directly publish data onto the IOTA Tangle on behalf of the energy asset. The device

registers its identity (public key) on the IOTA Tangle thus allowing it to encrypt and sign all the generated messages for a specific party. This would then expand the verification workflow to encompass further parts of the data chain as shown in the following figure. Further documentation can be found in Documentation (IOTA enabled Hardware)[22]. Additional information and further developments will be given in the forthcoming Deliverables D5.3: Campus Microgrid Model Prototype and D5.5: Energy Trading Market Demonstration.

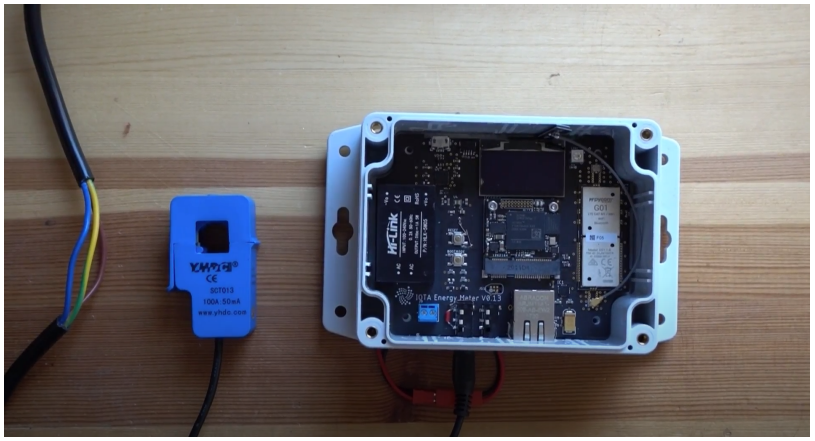

*Figure: IOTA asset module with integrated meter (Source: D2.7 (Livik et al., 2021))*

### 6.2.3 eMaaS IOTA prototype (POC)

The +CityxChange eMaaS system (Skoglund et al., 2020)was planned to include IOTA technology for payments and some data with charging stations in order to provide a seamless EV charging experience along journeys, including rental cars. This has been adapted from the ideal view to the realities, which includes the limitation of IOTA to the backend, not using an IOTA wallet directly in the eMaaS app, and (currently) not using the IOTA currency for payments. To complement the implemented eMaaS system, instead a prototype standalone platform was provided for multimodal journey planning and seamless payment across multiple journey providers using IOTA wallets and tokens. More details are described in the respective report (Skoglund et al., 2020).

The use cases on eMaaS (Sections 5.2 and 5.3, see also D1.2 (Petersen et al., 2021)) showcase the current demo architecture. The full IOTA-enabled SW architecture envisioned below has not (yet) been implemented. The diagrams below summarise IOTA's suggested transportation system architecture with deeply integrated payment and billing.

[22] https://drive.google.com/file/d/14JeJH4cBmZUnX4kVO24Zmro3LwbRoPCk/view?usp=sharing

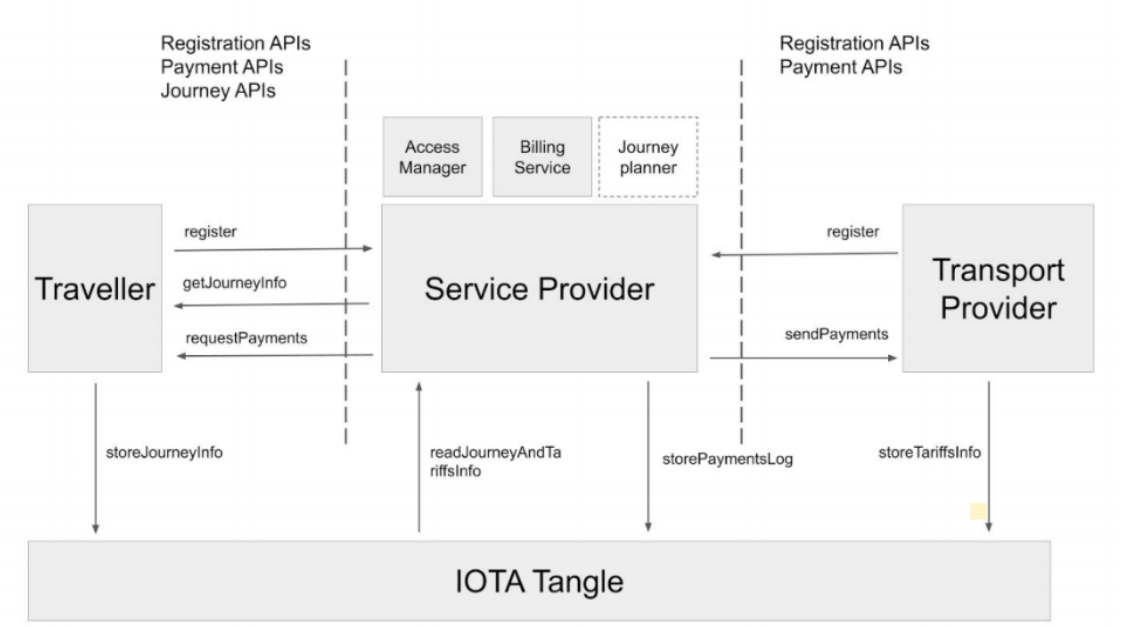


*Figure: IOTA Tangle, related to API no. 4, also APIs from no.13 to 20 (cf. Appendix A), and related to the eMaaS use case in D1.2* (Petersen et al., 2021) *and Section 5.3*

In this system, a Transport Provider registers to a Service Provider by providing its IOTA Wallet. A Traveller can register to the Service Provider with its IOTA Wallet and can obtain information about possible travel options satisfying its preferences and considering multimodal transports. Once a journey is planned the traveller receives a QR code for its mobile app for each transport provider involved in its journey. At the start of each leg the traveller scans the corresponding QR code and at the completion of a given leg the due amount is deducted by its IOTA wallet and transferred to the IOTA Wallet of the corresponding transport provider without extra fees and delays. This is also log for Auditability in the IOTA ledger. Further details on this can be found in the Documentation (Tech specs & requirements).[23]

## 6.3 Assessment of DLT use in +CityxChange

The DLT/IOTA integration could be shown within the data integrity and verification service integrated into Powels marketplace and within the location logging service for 4C and their eMaaS solution.

A number of practical issues currently hinder the full deployment within the real-life situation in the PEBs outside of fully controlled environments. This includes regulatory aspects due to payments as well as difficulty to integrate with legacy systems or hardware or market driven partners.

[23] https://drive.google.com/file/d/1eCDRZ8kL9gxviDzmq3USAyI4hdf_RCsZ/view?usp=sharing

In order to demonstrate the innovation potential of the IOTA ledger and feeless token, stand-alone demonstrations have been implemented. Within the eMaaS solutions the idea was to demonstrate how using the ledgers' auditability features and a digital token that does not incur transaction fees makes for a more efficient and trusted distribution of payments across a multi-stakeholders ecosystem of transport providers and a seamless experience for travellers who do not have to deal with multiple payment systems at once (and to provide an auditable data trail with controlled access and data integrity). Within the energy marketplace IOTA acts as a data transfer layer and/or a parallel data verification layer, thus demonstrating the potential for real world deployment of either scenario. DLT/IOTA as a means of data transfer is possible, advantageous and likely to be utilised in more solutions in the future. It can bring additional benefits to systems transferring data using traditional models, so that "assumed" trust is added to all such data transfers.
On the main blockers for full integration of DLT solutions, many partners point to the above issues , while in the IOTA view, also lack of acceptance and understanding and lack of willingness to migrate from traditional "tried and tested" data transfer models to a novel DLT based model is seen as a risk. IOTA ran and offered workshops to build trust in the technology, however, IOTA found that a large friction still existed around actual implementation of DLT services into pre existing technology stacks. IOTA wishes to educate and push understanding of DLT services in any way they can to speed up adoption of this new technology.

On an overall perspective, it could be demonstrated within selected prototypes how data exchange is possible with IOTA, but a full integration in existing buildings, cars, systems, etc. is still a challenge that is slowly being addressed.

# 7 Reflections

This section summarises some reflections and the main learning from the work presented in this deliverable.

**1) DIIF, to enable Data Integration and Interoperability for partners, is the main learning outcome.** Partner feedback and points *learnt* together with partners during the RI&D and design process (as in section 4.1) and the practical collaborative work on interoperability, which has culminated in DIIF framework (and *sub-innovations* reported in sections 3 and 4). Among the feedback from partners used for DIIF design was for example: slow pace of communications exchange made it difficult for partners to collaborate in the ICT ecosystem, which resulted in introducing the cross-partner requirements table.
Partners have been active in the design science process in all levels, while their work has focused on the practical aspects of level 1.
An anonymous survey was put forward at the end of this task to get more anonymous and unbiased feedback. The question "how good/bad is the +CxC DIIF in helping with 'ensuring Data Integration and Interoperability' for the consortium" was answered by 7 respondents, under a Likert scale from "very good" to "very bad". 2 (29%) responded with "average" and 5 (71%) answered "good". This shows a slightly positive-leaning feedback of the respondents on the usefulness of the framework, but also shows that there are limitations to the work, the interaction, and the structure.

**2) Best Practices must be actively used in an Agile way:** partners encouraged the introduction of simple tools (the simple tools appear in Level 1 of the framework) and all in all (indirectly) pointed to the necessity of an Agile approach (which the DIIF is based on) as also followed in the project as a whole. While an innovative framework which is best-practice-rich has been suggested, it is reflected in Level 1 of DIIF that the complex challenge of ensuring data integration and interoperability can be *gradually* overcome in a Smart City consortium such as +CityxChange by taking an Agile approach and providing (by the authors) and implementing/using (by partners) relevant framework and tools in an evolutionary and agile way.
**Starting with simple and easy to use tools/designs** before going to more complex frameworks and higher levels was a strong feedback from partners and reflects the literature. Implementing Project and Program Management best practices such as *PRINCE2 Agile and MSP* can accelerate this.

**3) Interoperability is a project-level crosscutting task.** In the structure of the project, specific implementation of the framework and the combination of interoperability issues has been slow or distributed and subject to individual developments instead of a common approach. Implementation of the framework was rather a matter of advice than guaranteed

action. Results were still achieved, but common standard development was not taken up as expected. In many cases, APIs were developed between two partners, while remaining invisible to the rest of the consortium. Better tool support may help. It also remained difficult to obtain common data models, this could be made clearer upfront to partners to make all data models and APIs explicit, similar to the API mandate requesting all communication between partners to go through only defined APIs[24], though this would be hard to achieve in an extremely distributed project and is part of ongoing research. One option to solve this in the future could be to raise the issue to a cross-cutting project task and to operate an interoperability-active project/portfolio management office, to accelerate the use-feedback-redesign loop.

**4) API Catalogue entries (~70)** show the reality of consortium APIs reported and are among the learnings, and are used to reflect the ICT ecosystem in the EA Use Cases in D1.2.

**Main Impact Brief Summary:** through *action research / design science* (Järvinen, 2005) together with the partners (benefitting from their feedback in numerous meetings), we have developed a *data integration and interoperability framework* (evolving maturity model) *for +CityxChange* and similar smart city projects/programs, we have introduced *three levels* of that maturity model together with many relevant (if not necessary) rules and sub-rules and standards and best practices, plus the simple and easy to use framework and tools for *level one (API Catalogue and Cross-partner Data-related Requirements Table)*. This process has benefitted from partner feedback. Partners are enabled, through the framework and data meetings, to ensure data integration and interoperability. The framework has been communicated and support given to help with framework understanding and implementation.

The innovations we have put forward are summarised in a table in Appendix J.

[24] https://nordicapis.com/the-bezos-api-mandate-amazons-manifesto-for-externalization/

# 8 Conclusion

This report addresses *data integration and interoperability* through developing a lightweight +CityxChange Data Integration and Interoperability Framework. The main contributions of this work is the DIIF, to support interoperability within and across the ICT ecosystems developed in the +CityxChange project, and the practical development work of partners. The DIIF consists of three levels of support: Level 1 enhances earlier work on interoperability frameworks and provides guidance and tools to support interoperability at the technical, business, organisational and governance levels of a city; level 2 provides project portfolio management support for interoperability projects as well as standards and best practices to achieve interoperability; and level 3 provides support for customisation of the standards and best practices available from level 2 for the specific activities.

An Agile rationale has been realized to manage and help with *ensuring* data integration and interoperability progress in the consortium, where easy to use tools and frameworks such as an API Catalogue and a Cross-partner Data-related Requirements Table have been designed and introduced to the partners for use, and usage of these have been followed up with the partners in several face-to-face and e-meetings. Standards, best practices and frameworks have also been introduced. For example, the development of the API Catalogue (Level 1 of the framework) is done, and an API specification format has been selected with the project partners. Then, together with partners, metadata and specification for APIs have been added to the API catalogue (see Section 5 and Annex A). As the partners are active in the early stages of the implementation of their solutions, they have not yet implemented Level 2 (and 3) at the time of writing this report, but communication of these levels and applying partner feedback to the *design* of DIIF has been done (reported from section 3 onwards, summary in section 3.2). For example, enhanced designs for the API Catalogue and also development of an advanced framework for DLT Service Assessment would be next steps in framework development and customisation.

Regular data meetings with partners have been a valuable part of this task. Due to the agile and iterative development approach adopted, the work described in this report has been enriched by the timely and relevant feedback received from the partners in the project while developing their own systems and APIs. The feedback from the partners has contributed to improved designs and the applicability of the DIIF.

The main lessons learned have been the importance of interactions with partners and collaborating and working with them in the development of the solutions. Starting with simple and easy tools and designs is important; e.g. API Catalogue and the Cross-partner Data-related Requirements Table. These tools can also serve as collaboration support among the partners that collaborate to develop solutions and ICT ecosystems.

In the future, the ICT ecosystem of the +CityxChange project would be enhanced by addressing further implementation of levels 2 and 3 of the DIIF. This work will be supported by sharing the solutions implemented by the Lighthouse cities and by providing guidelines and recommendations to the Follower Cities, based on the experience and feedback from the Lighthouse cities.

For replication, implementation processes should follow the processes laid out in Section 4, take inspiration and guidance from the cases described in Section 5, link the existing city systems and a data set mapping with the newly developed PEB systems, and ensure that different solution providers have an arena in which to jointly discuss ICT integration and data exchange. That should cover technical areas, tools and data, as well as organisational aspects.

## 8.1 Future Work and maturity improvements

This current deliverable aims to achieve data integration and interoperability for the ICT ecosystem developed in D1.2; including software platforms and tools, systems and services, data repositories, and IoT devices. This task further ensures agreement on open standards between service providers involved in demo projects for interoperability and supports data flow between partners and towards the KPI collection into the monitoring platform.

In addition, the work presented in this deliverable describes the proposed the DIIF (Data Integration and Interoperability Framework) with 3 layers. The levels are designed to incorporate technical interoperability and complement it with the complexity of the organisational, business and governance needs for interoperability. Customisations and maturity assessments can form a unified blueprint and help to achieve full interoperability within smart cities. In this project initial work was done with partners to demonstrate customisations of best practices, API catalogue enhancements, and lessons learned, limited to the scope and use cases presented in this document.

With continuing implementation, open standards for data vocabularies and data models in the smart city domain employed in the lighthouse cities will be further examined in future as these standards can be useful to follower cities for data integration and interoperability development in their cities.

Additionally, further work is required and expected to mature the solution to level 3 of the DIIF framework and expand the tools and templates.

- As shown with the API catalog, Level 1 extends the ICT development process supporting the overall development of ICT systems and the Enterprise Architecture

(EA) across the Ecosystem and is more concerned with computer and technical interoperability.

- Aiming to increase quality of the development and maintainability, and fostering better development and integration processes, Level 2 provides selected support and guidance for the development, documentation and management of APIs.
- Level 3 focuses on customisation of resources, suggesting innovations and achieving automation and optimization that includes for example automatic API publishing, discovery, monitoring and management.

Finally, further research would be required to detail the DIIF levels with subtasks, formalise and expand the approaches and develop detailed maturity metrics that are able to capture the various levels of interoperability. The API catalog could be expanded and automated for API discovery, traceability and versioning representing API lifecycle management. Further avenues for research are establishing API contracts and API governance[25] approaches.

### 8.1.1 Future Direction: DLT-enabled Service Assessment Model

To go one step further from broad requirements, DLT services should be assessed and governed well. This assessment can complement the requirements analysis. As DLTs and different data storage and exchange mechanisms are increasingly explored, the existing assessment models and metric systems may seem inadequate for providing a high-fidelity and thorough picture of service assessment and analytics. From the limited number of umbrella standards for assessment of DLTs, we highlight a United Nations standard (Ibáñez Jiménez et al., 2019) and its relevant assessment specifications developed by ITU for regulating and assessing DLTs (Yang et al., 2019). This takes into account views from service provider, service user, and regulator.

For this purpose, previous work of two authors of this report on general service assessment through a base cloud analytics model (Shams et al., 2021) is being enhanced and customised according to +CityxChange partner feedback and the gained experience from applying the previous model to the wider scope of DLT services for the smart city:

1. Generalisation from cloud analytics to service analytics: non-cloud stakeholder APIs such as DLT to be included
2. Improvement of the process loop
3. Metric categories and attributes are extended according to new feedback and experience
4. Customisation/extension of metrics and attributes for the assessment of Smart City ICT services to better accommodate DLT inclusion

[25] https://nordicapis.com/api-improvement-proposals-googles-take-on-the-api-style-guide/

The updated design of the measurement model addresses a number of existing and emerging quantitative analysis challenges to arrive at a new service-oriented technical assessment frameworks customised for the purpose of DLT service use. It will include project considerations and input from IOTA, and considerations of the ITU DLT standard. This work is under development and is planned to be published as a forthcoming journal paper.

# References

Achilleos, A., Yang, K., & Georgalas, N. (2008), November. A model driven approach to generate service creation environments. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference* (pp. 1-6). IEEE.

Ahlers, D., Brigg, D., Karatzoudi, K., & Wyckmans, A. (2020). D11.16 Data Management Plan, *+CityxChange Deliverable*, +CityxChange Project. https://cityxchange.eu/knowledge-base/d11-16-data-management-plan-3/

Ahlers, D., Wienhofen, L., Petersen, S.A., & Anvaari, M. (2019). A Smart City Ecosystem enabling Open Innovation. *19th International Conference on Innovations for Community Services (I4CS 2019)*. CCIS, vol 1041. Springer. 2019. doi: 10.1007/978-3-030-22482-0_9.

Alawadhi, S., Aldama-Nalda, A., Chourabi, H., Gil-Garcia, J.R., Leung, S., Mellouli, S., Nam, T., Pardo, T.A., Scholl, H.J., & Walker, S. (2012), September. Building understanding of smart city initiatives. In *International conference on electronic government* (pp. 40-53). Springer, Berlin, Heidelberg.

Amundsen, M. (2020). Design and Build Great Web APIs. *Pragmatic Bookshelf*.

Antolić, M., Wright, S., Mulugeta, D., Fullam, J., Stewart D. (2021). D2.6: Framework for Community Grid Implementation, *+CityxChange Deliverable*, +CityxChange Project.Baskerville, R. (2008). What design science is not. *European Journal of Information Systems*, 17(5), 441–443.10.1057/ejis.2008.45

Berre, A.J., Elvesæter, B., Figay, N., Guglielmina, C., Johnsen, S.G., Karlsen, D., Knothe, T., & Lippe, S. (2007). The ATHENA interoperability framework. In *Enterprise interoperability II* (pp. 569-580). Springer, London.

Bertelsen, S., Livik, K., & Myrstad, M. (2019). D2.1: Report on Enabling Regulatory Mechanism to Trial Innovation in Cities, *+CityxChange Deliverable*, +CityxChange Project.

Bhatt V., Brutti A., Burns M., & Frascella A. (2017). An Approach to Provide Shared Architectural Principles for Interoperable Smart Cities. In: *Computational Science and Its Applications. ICCSA 2017*. LNCS, vol 10406. Springer. https://doi.org/10.1007/978-3-319-62398-6_29

Calzada, I. (2019). Report on the City-to-City-Learning Programme: the replication strategy in Replicate EU-H2020-Smart Cities and Communities (SCC) Lighthouse Project, DOI: 10.13140/RG.2.2.21412.83843/2.

Chiprianov, V., Alloush, I., Kermarrec, Y., & Rouvrais, S. (2011), July. Telecommunications Service Creation: Towards Extensions for Enterprise Architecture Modeling Languages. In *ICSOFT* (1) (pp. 23-28).

Chiprianov, V., Kermarrec, Y., & Rouvrais, S. (2011b), June. Practical meta-model extension for modeling language profiles. An enterprise architecture modeling language extension for telecommunications service creation.

Dahlen, K.E., Livik, K., Purshouse, N., & Antolic, M. (2020). D2.2: Toolbox for design of PEB including e-mobility and distributed energy resources, *+CityxChange Deliverable*, Confidential, +CityxChange Project.

EIF Contributors (2020). European Interoperability Framework (EIF). *Joinup Network Website*, European Commission, URL: https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/eif-european-interoperability-framework-0

EIRA contributors (2019). European Interoperability Reference Architecture (EIRA) release v3.0.0, *Joinup Network website*, European Commission.

Exner, J.P. (2016). The ESPRESSO-Project – A European Approach for Smart City Standards. In *International Conference on Computational Science and Its Applications* (pp. 483-490). Springer, Cham. https://doi.org/10.1007/978-3-319-42111-7_38

FitzPatrick, G.J., & Wollman, D.A. (2010), July. NIST interoperability framework and action plans. In IEEE PES General Meeting (pp. 1-4). IEEE.

Faber, A., Hernandez-Mendez, A., Rehm, S.V., & Matthes, F. (2018), March. An Agile Framework for Modeling Smart City Business Ecosystems. In *ICEIS* (2) (pp. 39-50).

Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. IETF. RFC 6455.

FIWARE Contributors (2016), FIWARE Data Model for Key Performance Indicators, Fiware Project, Core Platform for the Future Internet, Private Public Partnership Project (PPP), *FIWARE website*, ongoing development on https://www.fiware.org. URL: https://fiware-datamodels.readthedocs.io/en/latest/KeyPerformanceIndicator/doc/spec/index.html

FIWARE Foundation (2021), FIWARE Catalog. *FIWARE website*, URL: https://www.fiware.org/developers/catalogue/ (Accessed: 21 Jan 2021)

Fitzgerald, H., Burón García, J., Sánchez Mora, M. (2020). D3.6: Framework for DPEB Innovation Labs, +CityxChange Deliverable, +CityxChange Project. https://cityxchange.eu/knowledge-base/d3-6-framework-for-dpeb-innovation-labs/

Gottschalk, P. (2009). Maturity levels for interoperability in digital government. *Government Information Quarterly*, 26(1), pp.75-81. https://doi.org/10.1016/j.giq.2008.03.003

Hevner, A.R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2), p.4.

Ibáñez Jiménez, J.W., Chuburkov, A., Levashov, A., Khramtsovsky, A., Khramtsovsky, N., Marshalko, G.B., Arribas, I., Ibáñez, J., Wack, J., Erbguth, J., Neves, L.P., Lyons, P.A., Payen, P., Griffin, P.H., O'Brien, R., & Hu, T. (2019). Distributed Ledger Technology Regulatory Framework. *Technical Report FG DLT D4.1*, The International Telecommunication Union (ITU). United Nations, August 2019. https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d41.pdf

IEEE dictionary contributors (1990). IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries. *IEEE (Institute of Electrical and Electronics Engineers)*, New York: IEEE; 1990.

Järvinen, P. (2005). Action research as an approach in design science. University of Tampere, Report D-2005-2 (EURAM conference).

Johannesson, P., & Perjons, E. (2014). An introduction to design science. Springer.

Kerrigan, R., Purshouse, N., De Donatis, L., & Neu O. (2020). D4.1: Limerick DST (Integrated Modelling and Decision Support Tool) including training manuals/videos, *+CityxChange Deliverable*, +CityxChange Project.

Medjaoui, M., Wilde, E., Mitra, R., & Amundsen, M. (2018). Continuous API Management: Making the right decisions in an evolving landscape. O'Reilly Media.

Livik, K., Danielsen, S., & Nati, M. (2021). D2.7: Local DPEB trading market demonstration tool, *+CityxChange Deliverable*, +CityxChange Project. https://cityxchange.eu/knowledge-base/d2-7-local-dpeb-trading-market-demonstration-tool/

Office of Government Commerce GB (2007). Managing successful programmes (MSP). TSO.

Ojo, A., Curry, E., Janowski, T., & Dzhusupova, Z. (2015). Designing next generation smart city initiatives: The SCID framework. In *Transforming city governments for successful smart cities* (pp. 43-67). Springer, Cham.

Purshouse, N., De Donatis, L., Neu, O., & Kerrigan, R. (2021). D1.4: Demonstration of the +CityxChange Integrated Modelling Platform, *+CityxChange Deliverable*, Confidential, +CityxChange Project.

Petersen, S.A., Bokolo, A.J., Ahlers, D., Krogstie J., Shams, A., Helfert, M., Alloush, I., & Pourzolfaghar, Z. (2021). D1.2: Report on the Architecture for the ICT ecosystem, *+CityxChange Deliverable*, +CityxChange Project. https://cityxchange.eu/knowledge-base/d1-2-report-on-the-architecture-for-the-ict-ecosystem/

Pimenta de Miranda, W. (2019). Towards Open & Transparent Cities: An IOTA Ecosystem Report. In *IOTA Foundation website*, accessed Sep. 2019. URL: https://files.iota.org/comms/IOTA_smartcityreport.pdf

Rood, D. (2019). D7.4: Monitoring and Evaluation Dashboard, *+CityxChange Deliverable*, +CityxChange Project. https://cityxchange.eu/knowledge-base/monitoring-and-evaluation-dashboard/

Saleem, M., Chung, P.W., Fatima, S., & Dai, W. (2014). A cross organisation compatible workflows generation and execution framework. *Knowledge-Based Systems*, 56, pp.1-14.

Shams, A., & Kermanshah, A. (2018). *Total Change Management*, Supreme Century, URL: https://www.amazon.com/Total-Change-Management-Persian-Armin/dp/1939123739

Shams, A., Sharif, H., & Helfert, M. (2021). A Novel model for cloud computing analytics and measurement. *Journal of Advances in Information Technology*, Vol. 12, No. 2, pp. 93-106, May 2021. doi: 10.12720/jait.12.2.93-106.

Shams, A., Sharif, H., & Kermanshah, A. (2017), June. Holistic Change Management: Importance and methodological challenges. In *2017 IEEE Technology & Engineering Management Conference (TEMSCON)* (pp. 272-276). IEEE.

Shen, C., & Pena-Mora, F. (2018). Blockchain for cities—a systematic literature review. *Ieee Access*, 6, pp.76787-76819.

Skoglund, T.R., Main, D., Eljueidi, M., & Nati, M. (2020), D2.5: Seamless eMobility system including user interface. *+CityxChange Deliverable D2.5*, +CityxChange Project, 2020. https://cityxchange.eu/knowledge-base/d2-5-seamless-emobility-system-including-user-interface/

The Open Group (2011). The Open Group Architecture Framework TOGAF Version 9.1. 2011.

Tóth, K., & Smits, P. (2007). Infrastructure for spatial information in Europe (INSPIRE): from cartography to spatial objects and network services. In *Proceedings of the 23rd Cartographic Conference–Cartography for everyone and for you* (pp. 1-11).

van der Aalst, W.M. (1999). Process-oriented architectures for electronic commerce and interorganizational workflow. *Information systems*, 24(8), pp.639-671.

Wyckmans, A., Vandevyvere, H., Gohari, S., Nielsen, B.F., Driscoll, P., Ahlers, D. (2019). D9.1: Framework for intra-project collaboration, *+CityxChange Deliverable*, +CityxChange Project. https://cityxchange.eu/knowledge-base/framework-for-intra-project-collaboration/

Yang, B., Kai, W., & Hu, R. (2019). Assessment criteria for distributed ledger technology platforms. *Technical Specification FG DLT D3.3*, The International Telecommunication Union (ITU), under United Nations, August 2019. https://www.itu.int/en/ITU-T/focusgroups/dlt/Documents/d33.pdf

Ziemann, J., Matheis, T., & Freiheit, J. (2007). Modelling of cross-organizational business processes-current methods and standards. *Enterprise Modelling and Information Systems Architectures-An International Journal*: Vol. 2, Nr. 2.

# Appendix

## Appendix A – API Catalogue

The following table is an extract of the API catalogue, containing the API list, description, metadata, with sensitive or confidential information removed. More information regarding each API appears in the internal API Catalogue).[26] Details on the catalogue and the EA use case are discussed in Section 5 and outcomes given in Appendix B. This is the state at the time of writing.

[26] The API information is obtained from the partners and at the time of writing the report; some APIs are under development and some pending better documentation.

| No. | Owner | API Title | API Aim | Description | Spec. available | Detailed specs, code etc. | Standards, protocols, etc. mentioned |
|---|---|---|---|---|---|---|---|
| 1 | Reserved | Reserved | Reserved | Reserved (retired) | - | - | - |
| 2 | LCCC | LCCCC-cctv-locations-map | CCTV-Locations | CCTV location in form of coordinates and a number of properties (by Limerick City and County Council) - Static API to fetch information | Yes | Yes (code) | JSON, REST, standard geographical coordinates |
| 3 | FAC | GetKPI | Get KPI | Getting KPI information from the MERT (+CxC dashboards, showing KPIs provided by different partners, used by LCCC and other stakeholders) | Yes | Yes (code with details) | JSON, REST |
| 4 | IOTA | eMaaS-locationlogging-location | Location logging to ledger | To store location data for a new or existing trip, the mobile app sends appId, tripId and location data parameters to the cloud backend /location API. | Yes | Yes (code) | JSON, REST, HTTPS/TLS[2] |
| 5 | IOTA | eMaaS-locationlogging-claim | To retrieve location data in case of a claim | To retrieve location data in case of a claim for a selected trip, the mobile app sends appId and tripId parameters to the cloud backend /claim API. If the combination of appId and tripId is found, location data will be fetched from the Tangle, decrypted and sent to the mobile app. | Yes | Yes (code) | JSON, REST, HTTPS/TLS |
| 6 | Reserved | Reserved | Reserved | | - | - | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | TE | ForecastAPI | TE forecast to be provided to ABB via API | TE forecast to be provided to ABB via API | No (under dev.) | No | JSON, REST |
| 8 | IOTA | getVersion | Get the API details | Outputs the API details | Yes | Yes (code) | JSON, REST |
| 9 | IOTA | getDocs | Get the documentation for the API | Get the documentation for the API | Yes | partly (partial code) | XML, REST |
| 10 | IOTA | transportProviderRegister | Register a new transport provider in the system | Register a new transport provider in the system | Yes | Yes (code) | JSON, REST, HTTPS/TLS |
| 13 | IOTA | transportProviderTariffCreate | Create the tariffs for the transport provider. | Create the tariffs for the transport provider. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 11 | IOTA | transportProviderLogin | Login a transport provider | Login a transport provider. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 12 | IOTA | transportProvidersGet | Gets a list of the transport providers. | Gets a list of the transport providers. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 14 | IOTA | transportProviderTariffGet | Get the tariffs for the requested transport provider. | Get the tariffs for the requested transport provider. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 15 | IOTA | transportProviderPaymentUpdate | Update the payment details for the transport provider. | Update the payment details for the transport provider. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 16 | IOTA | userRegister | Register a new user in the system. | Register a new user in the system. | Yes | Yes (code) | JSON, REST, HTTPS/TLS |
| 17 | IOTA | userLogin | Login a user. | Login a user. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 18 | IOTA | userJourneyCreate | Create a journey for the user. | Create a journey for the user. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 19 | IOTA | userJourneysGet | Gets a summary list of all the users' journeys. | Gets a summary list of all the users' journeys. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 20 | IOTA | userJourneyEventCreate | Add an event to an existing journey | Add an event to an existing journey | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 21 | IOTA | userPaymentMethodCreate | Create a new payment method for the user. | Create a new payment method for the user. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 22 | IOTA | userPaymentMethodGet | Gets a summary list of all the users payment methods. | Gets a summary list of all the users payment methods. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 23 | IOTA | userPaymentMethodUpdate | Update a payment method for the user. | Update a payment method for the user. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 24 | IOTA | userPaymentMethodDelete | Delete a payment method for the user. | Delete a payment method for the user. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 25 | IOTA | userPaymentMethodFundsAvailable | Check a users payment method to see if funds are available. | Check a users payment method to see if funds are available. | Yes | Yes (code) | JSON, JWT, REST, HTTPS/TLS |
| 26 | IES | iSCAN (part of DST) | To allow configuration and use of a time series database for building monitoring and analysis. | Allows configuration and use of a time series database for building monitoring and analysis. IES Tool consisting of iVN (API no. 69) and iSCAN (API no. 26) provides building level and demand curve calculation, load curve and also grid calculation. iSCAN is a web-based tool from IES which is used to import, manage and analyze operational (time-series) data from various sources in a single platform. Measured building energy consumption data from smart meters, or simulated energy consumption data from the IESVE, can be uploaded to iSCAN. This data can then be imported into the iVN tool as a profile asset to be assigned to a building in the iVN model. | Yes | Yes (rather comprehensive, no code, internal info links are provided) | JSON, REST, HAL, export formats such as CSV, EPS, APM, HTTPS/TLS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 27 | FourC AS | TTC Geo API | To output geographical data, which can be displayed and interacted on maps. | The TTC Geo API is an API designed to allow for the best possible integration with different map solutions. The API is mainly designed to output geographical data, which can be displayed and interacted on maps. The objects themselves are data rich, but it's up to each individual user of the API to utilize the information that is made available. The API is based on outputting the data as GeoJSON objects (RFC 7946). The GeoJSON specification lets the user easily take the data out as a standardized object format and display it on a map. GeoJSON is also very frequently supported natively by map solutions to display the objects. The API is based on GraphQL, a modern alternative to the more traditional REST. [...] GraphQL standard also supports subscription based requests over Websockets. The API is very focused in utilizing this concept, as the data is dynamic and changes over time[...] | Yes | Yes (code and details) | JSON (GeoJSON), GraphQL, Websocket, HTTPS/TLS (querying and subscription) |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 28 | Trondheim City Bikes | Trondheim City Bikes | Trondheim City Bikes | Trondheim City Bikes - Feeds 4C (feeds partner API) | Yes | Yes (code with details) | JSON, REST, Norwegian License for Public Data (NLOD) 2.0, HTTPS/TLS |
| 29 | Nobil | Nobil (charging stations for EVs) | Charging stations for EVs | Charging stations for EVs - Feeds 4C (feeds partner API) | Yes | Yes (code with details) | Websocket, JSON, HTTPS/TLS |
| 30 | Entur | Entur (public transport, bus, train, boat, etc.) | Public transport, bus, train, boat, etc. | Public transport, bus, train, boat, etc. - Feeds 4C (feeds partner API) | Yes | Yes (code with details for the API Package) | REST, JSON, ISO 3166-1 alpha-3 country code, HTTPS/TLS, XSD XML, GTFS, NeTEx |
| 31 | Rent-Centric | Rent-Centric | Rent-Centric | Rent-Centric - Feeds 4C (feeds partner API) | Yes | Yes (code) | REST, JSON, HTTPS/TLS |
| 32 | National Road Database | National Road Database | National Road Database | National Road Database - Feeds 4C (feeds partner API) | Yes | Yes (code with details) | REST, JSON (default), XML, HTTPS/TLS |
| 33 | AVINOR | AVINOR (Norwegian Airports) | AVINOR (Norwegian Airports) | AVINOR (Norwegian Airports) - Feeds 4C (feeds partner API) | Yes | Yes (code with details) | REST, JSON, XML (for flight data), IATA Code, ISO 8601 Time |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | Format, HTTPS/TLS |
| 34 | Adbshub.org | Adbshub.org (real time aircraft positions) | Real time aircraft positions | Real time aircraft positions - Feeds 4C (feeds partner API) | Yes | Yes | REST, JSON, data format provided via TCP connection is the popular SBS (30003) format |
| 35 | Trøndertaxi | Trøndertaxi | Trøndertaxi | Trøndertaxi - Feeds 4C (feeds partner API) | partly | No | Probably REST, JSON/XML |
| 36 | AtB public transport authority | AtB (for bus location) | Bus location | Bus location - Feeds 4C (feeds partner API) | Yes | Yes (code with details) | REST, JSON, GTFS, Apache License Version 2.0, HTTPS/TLS, UTF-8 |
| 37 | Værnes-Ekspressen | Trondheim Airport Express Buses | Trondheim Airport Express Buses | Trondheim Airport Express Buses - Feeds 4C (feeds partner API) | partly | partly (standard referenced and linked with code and details) | REST, JSON, XML, XSM, BODS, SIRI-VM |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 38 | MPOWER | SLU API | Supply raw voltage, current and power factor data from prosumer SLUs | Supply raw voltage, current and power factor data from prosumer SLUs, used by SLU devices for readings, LoRa, trade requests | Yes | Yes (code with details) | REST, JSON, Mutual SSL Authentication |
| 39 | MPOWER | Operator API | internal use by MPower employees | internal use by MPower employees | No | No (internal, under dev.) | REST, JSON |
| 40 | MPOWER | Prosumer API | used by prosumers for trading, billing control, home management | used by prosumers for trading, billing control, home management | partly | No (internal, under dev.) | REST, JSON, possibly IOTA's TANGLE |
| 41 | MPOWER | Report and Partner API | anything (report) MPower needs to provide to partners | anything (report) MPower needs to provide to partners | partly | No (internal, under dev.) | REST, JSON |
| 42 | Powel | MarketPlatformData | Market Platform Data | Allows sharing of time series data regarding energy forecast, plans, transactions, etc. | Yes | Yes (code) | REST, JSON |
| 43 | TK | TK-KPIs | Providing TK KPIs | Contains KPIs from Trondheim municipality, to feed the FAC's dashboard software system (API no. 3 is relevant) | Yes | Yes (code and brief details) | REST, JSON, HTTPS/TLS |
| 44 | IOTA | AssetInitiation | Remote Initialization of IOTA asset module | Remote asset initialization or configuration update. The payload is sent unencrypted, but the asset accepts the payload only when the provider UUID matches the internal UUID of the device, which is only known to the owner of the device. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 45 | IOTA | AssetContract | Communicate energy match contract details | After a match between energy offer and energy request was found by the Bid Manager, and the contract between energy provider and energy requester is established, this endpoint is used to communicate contract details. The request payload consists of the offer transaction object, request transaction object, and the contract ID. Both contract partners, energy provider and energy requester, will receive the same /contract request. As a result of the /contract request the energy provider can immediately start energy provision to the consumer. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |
| 46 | IOTA | AssetPayment | Check and make payment for energy trade | Once energy provision is finished, and energy provision confirmation sent to the Marketplace, the energy requester will receive the /payment request from the Marketplace. Encrypted payload consists of the contract transaction object with updated status and timestamp. As a result of the /payment request the energy requester verifies if the request is valid by comparing a number of transaction parameters with the data stored in the local DB and in the corresponding MAM channel. If the payment request is valid, it will be added to the payment queue and processed together with other payment requests. See "Payment processing" topic. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 47 | IOTA | AssetPaymentsent | Confirms to the energy provider the payment has been sent and provides payment details | Once the energy requester reports successful payment processing to the Marketplace, the energy provider will receive the /payment_sent request from the Marketplace. Encrypted payload consists of the payment request transaction object with updated status and timestamp. As a result of the /payment_sent request the energy provider logs the transaction into the local DB and verifies payment in defined time intervals. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |
| 48 | IOTA | AssetPaymentconfirmed | Update energy provider once payment is confirmed | Encrypted payload consists of the payment confirmation transaction object with updated status and timestamp. As a result of the /payment_confirmed request the energy requester logs the transaction into the local DB and communicates to the User module. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 49 | IOTA | AssetPaymentcancel | Cancel pending payments | Pending and unpaid transactions that remain unpaid when the max. allowed waiting time is elapsed, can be cancelled. Both asset types can receive /cancel requests from the Marketplace. Encrypted payload consists of the transaction object with updated status and timestamp. As a result of the /cancel request the asset logs the transaction into the local DB and communicates to the User module. Energy providers will additionally unreserve energy that was reserved for the offer. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 50 | IOTA | AssetPaymentclaim | Claiming unpaid payments | Energy providers can issue claims for transactions that remain unpaid after 3 payment reminder requests. Energy consumers can receive /claim requests from the Marketplace. Encrypted payload consists of the transaction object with updated status and timestamp. As a result of the claim, energy consuming devices can be excluded from the Marketplace, and a penalty can be issued to the device's owner. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 51 | IOTA | AssetProduceenergy | Creating energy offer | /produce request, available energy amount is updated in the internal DB. If the total available energy exceeds the minOfferAmount value, a new energy offer is created and sent to the marketplace. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 52 | IOTA | AssetConsumeenergy | Creating energy request | /consume request, available energy amount is updated in the internal DB. If the total available | Yes | Yes (code) | REST, JSON, Possibility of |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | energy drops below the minOfferAmount value, a new energy request is created and sent to the marketplace. | | | using Tangle/HTTPS/ TLS/Base64 |
| 54 | IOTA | MarketplaceOffer | Energy producing assets issue energy offers and communicate them to the marketplace. | Energy producing assets issue energy offers and communicate them to the marketplace. The payload consists of the offer transaction object. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /offer call, the payload is decrypted, signature is verified, then the offer transaction is stored in the internal database of the Marketplace and communicated to the Bid Manager to find a matching request transaction. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |
| 53 | IOTA | MarketplaceRegister | Initial asset registration with Marketplace. | Initial asset registration with Marketplace. The request payload consists of the asset's data such as ID, public key, type, location, URL. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /register request, the payload is decrypted, signature is verified, then asset info is stored in the local database of the marketplace. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 55 | IOTA | MarketplaceRequest | Energy consuming assets issue energy requests and communicate them to the marketplace | Energy consuming assets issue energy requests and communicate them to the marketplace. The payload consists of the request transaction object. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /request call, the payload is decrypted, signature is verified, then the request transaction is stored in the internal database of the Marketplace and communicated to the Bid Manager to find a matching offer transaction | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 56 | IOTA | MarketplaceMatch | Contract ID is created, signed by the Marketplace and sent to both contract partners, energy provider and energy requester. | Response is sent from Bid manager to Marketplace. The payload is sent unencrypted, as Marketplace and Bid manager are located in the same network. As a result of the /match call, a payload containing the offer transaction object, request transaction object, and the contract ID is created, signed by the Marketplace and sent to both contract partners, energy provider and energy requester. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 57 | IOTA | MarketplaceProvision | The energy requester reports successful payment processing to the Marketplace. | The energy requester reports successful payment processing to the Marketplace. Encrypted payload consists of the payment transaction object with updated status and timestamp. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /payment_processing call, the payload is decrypted, signature is verified, then a payment confirmation is sent to the energy provider. The transaction is stored in the internal database of the Marketplace. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 58 | IOTA | MarketplacePaymentconfirmation | The energy provider reports payment confirmation to the Marketplace. | Encrypted payload consists of the payment transaction object with updated status and timestamp. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /payment_confirmation call, the payload is decrypted, signature is verified, then a payment confirmation is sent to the energy requester. The transaction is stored in the internal database of the Marketplace. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 59 | IOTA | MarketplacePaymentprocessing | The energy requester reports successful payment processing to the Marketplace. | The energy requester reports successful payment processing to the Marketplace. Encrypted payload consists of the payment transaction object with updated status and timestamp. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /payment_processing call, the payload is decrypted, signature is verified, then a payment confirmation is sent to the energy provider. The transaction is stored in the internal database of the Marketplace. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 60 | IOTA | MarketplaceCancel | Transactions that remain in its initial state after the max. allowed waiting time is elapsed, will be cancelled. | Transactions that remain in its initial state after the max. allowed waiting time is elapsed, will be cancelled. Both asset types can send /cancel requests to the Marketplace. Encrypted payload consists of the transaction object with updated status and timestamp. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /cancel request the Marketplace logs the transaction into the local DB. If the transaction already contains the contractID, the contract partner other than the sender of the request will be notified. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 61 | IOTA | MarketplaceClaim | Energy providers can issue claims for transactions that remain unpaid after 3 payment reminder requests. | Energy providers can issue claims for transactions that remain unpaid after 3 payment reminder requests. Encrypted payload consists of the transaction object with updated status and timestamp. The payload is sent encrypted with the public key of the Marketplace and signed by the asset. As a result of the /claim request, the Marketplace notifies the energy consumer and can optionally perform additional steps like excluding the asset from the marketplace or issue a penalty to the asset's owner. The Marketplace logs the transaction into the local DB. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |
| 62 | IOTA | BidmanagerOffer | New offer data received and processed by the Marketplace module is communicated to the Bid manager. | New offer data received and processed by the Marketplace module is communicated to the Bid manager. The payload is sent unencrypted, as Marketplace and Bid manager are located in the same network. As a result of the /offer call, Bid manager performs lookup for a matching request. If a match is found, a matching request is removed from the internal database, and both transactions are bundled and communicated to the Marketplace, otherwise the offer is stored in the internal database. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/ TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 63 | IOTA | BidmanagerRequest | | New request data received and processed by the Marketplace module is communicated to the Bid manager. The payload is sent unencrypted, as Marketplace and Bid manager are located in the same network. As a result of the /request call, Bid manager performs lookup for a matching offer. If a match is found, a matching offer is removed from the internal database, and both transactions are bundled and communicated to the Marketplace, otherwise the request is stored in the internal database. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 64 | IOTA | BidmanagerRemove | Called when the offer or request was cancelled. | Called when the offer or request was cancelled. As a result of the /remove call the offer or request is removed from the internal database. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 65 | IOTA | UserFund | Energy consumers can request wallet funding from the asset's owner | Energy consumers can request wallet funding from the asset's owner, if the wallet balance is not sufficient to pay for the consumed energy. The payload consists of the asset's wallet address, current balance, asset type and asset ID. The payload is sent encrypted with the public key of the asset's owner and signed by the asset. As a result of the /fund call, the payload is decrypted, signature is verified, then a token transfer is initiated from the asset's owner wallet to the asset's wallet. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 66 | IOTA | UserNotifyevent | All asset types notify their owners about the new offers/request and any change in the status of the existing requests. | All asset types notify their owners about the new offers/request and any change in the status of the existing requests. The payload consists of the transaction object. The payload is sent encrypted with the public key of the asset's owner and signed by the asset. As a result of the /notify_event call, the payload is decrypted, signature is verified, then the transaction is stored in the database for audit. | Yes | Yes (code) | REST, JSON, Possibility of using Tangle/HTTPS/TLS/Base64 |
| 67 | Central Statistics Office | csodata API package | Provides Census Data | Provides Census Data used by the Integrated Modelling and Decision Support Tool (DST) to analyze Socio Economic Data for Limerick. | Yes | Yes (Manual for 10+ APIs in API Package) | JSON, REST, UTF-8, GPL-3 License, HTTPS/TLS |
| 68 | LCCC and Environmental Protection Agency | Air Quality API | Provide data on Air Quality from OConnell Street Sensor | Provide data on Air Quality from OConnell Street Sensor (approximately every hour) Used In Limerick DST (Integrated Modelling and Decision Support Tool) (+CxC D4.1) | Yes | Yes | REST, JSON, HTTPS/TLS |
| 69 | IES | iVN (part of DST) | Communicates with iSCAN and other partner systems regarding PEB/PED models. The main purpose of the iVN is to assist utility operators, policy makers and | The main purpose of the iVN tool is to assist utility operators, policy makers and engineers with city or district level decision making with regards to the supply and management of natural and man-made resources and the devices that make that possible. Specifically, it can inform users of the current performance of various utilities and quantitatively predict the impact of changes to | Yes | Mostly(details linked code and brief spec under dev.) | REST, JSON |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | engineers with city or district level decision making with regards to the supply and management of natural and man-made resources and the devices that make that possible. | city-infrastructure. For example, adding a wind farm to a particular level of the electricity grid will generate electricity when it is windy, reducing the amount needed to be imported from abroad, reducing the carbon emission factors for the housing served by the wind farm and helping to populate storage devices with electricity in times of low demand; the effect on the whole city infrastructure is estimated naturally as a consequence of adding the wind farm to the network. iVN Communicates real time metered data with iSCAN, and communicates with other partner systems regarding PEB/PED models. | | | |
| 70 | Atb | tariff zones | List tariff zones | | Yes | Yes (link to code and software) | REST, JSON, HTTPS/TLS |
| 71 | FourC | Charger and Battery Simulator API | To provide car usage of charging stations, the charging parameters and the car battery status | The Charger and Battery Simulator (CBS) simulates a simple usage pattern of electric cars at electric chargers. The CBS was implemented as part of the +CityxChange project to enable other partners in the project to consume more or less realistic car usage of charging stations, the charging parameters and the car battery status, since no real data was available at the time. | Yes | Yes (code with brief details) | REST, JSON, HTTPS/TLS |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 72 | Third party - supported by ABB | ABB third party Protocols and APIs supported | Machine to Machine interface, for Energy Data in this case | ABB third party Protocols and APIs supported | Yes | Yes | HTTP (SOAP or REST) APIs, XML, Modbus TCP, Modbus RTU, OCPP, OPC-DA/UA, IEC 60870-5-104, SFTP |
| 73 | FourC | TTC API | | • SIRI VM interface for vehicle position. Currently, only PT vehicles are available. Other modes of transport can be added when standardized in a newer SIRI/transmodel version.<br>• NeTEx - route network data. Currently, supported for PT routes only.<br>• Bus stop position - TTC offers an export of bus stops, using concepts like Quay and StopPlace, as defined by Transmodel. Currently supports all stops defined in NVDB.<br>• Charger data - Positions and status/attributes, historical and real-time<br>• EV data - Core data like positions and status/attributes, historical and real-time<br>• Vehicle availability<br>• Journey description<br>• Road closures and status | Yes | Yes (no code) | REST, JSON, Transmodel (EU CEN TC278 standards) |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | • All data are stored historically. This means that the current status of an object and its attributes for a specific moment back in time can also be queried for. | | | |

# Appendix B – API Cases details in API Catalogue (linked to EA Cases)

This appendix lists the detailed recorded APIs per EA use case based on their structure in D1.2. Background is discussed in Section 5. The first cases are explained in detail with their EA visualisation from D1.2. The remainders are listed only from the linkage view, readers are directed to D1.2 to avoid duplication here.

## B.1 APIs for the Limerick Case

LCCC has the key role of facilitating the smart city efforts for Limerick, including data and interoperability aspects. As discussed in D1.2 (section 6.5 Overview of +CityxChange Use case for Limerick) the APIs used in Limerick are summarized in the following table.

| API number in catalogue | API name | API owner | API consumer | API description |
|---|---|---|---|---|
| 2 | lccc-cctv-locations-map | LCCC | LCCC | CCTV location in form of coordinates and a number of properties (by Limerick City and County Council) - Static API to fetch information |
| 3 | GetKPI | FAC | LCCC | Getting KPI information from the MERT (+CxC dashboards, showing KPIs provided by different partners, used by LCCC and other stakeholders) |
| 26 | iSCAN (part of DST) | IES | LCCC | To allow configuration and use of a time series database for building monitoring and analysis. |
| 27 | TTC Geo API | FourC | LCCC | To output geographical data, which can be displayed and interacted on maps. |
| 28 | Trondheim City Bikes | FourC | LCCC | Provide data on Trondheim City Bikes - Feeds 4C (feeds partner API) |
| 41 | Report and Partner API | MPower | LCCC | Provide information or anything (report) from MPower needs to provide to partners |

*Table: Description of the APIs in the limerick use case*

The Limerick EA Use Case shows how the different layers including infrastructures, technologies, APIs and VEs are used to achieve the context goals according to the +CityxChange Enterprise Architecture:

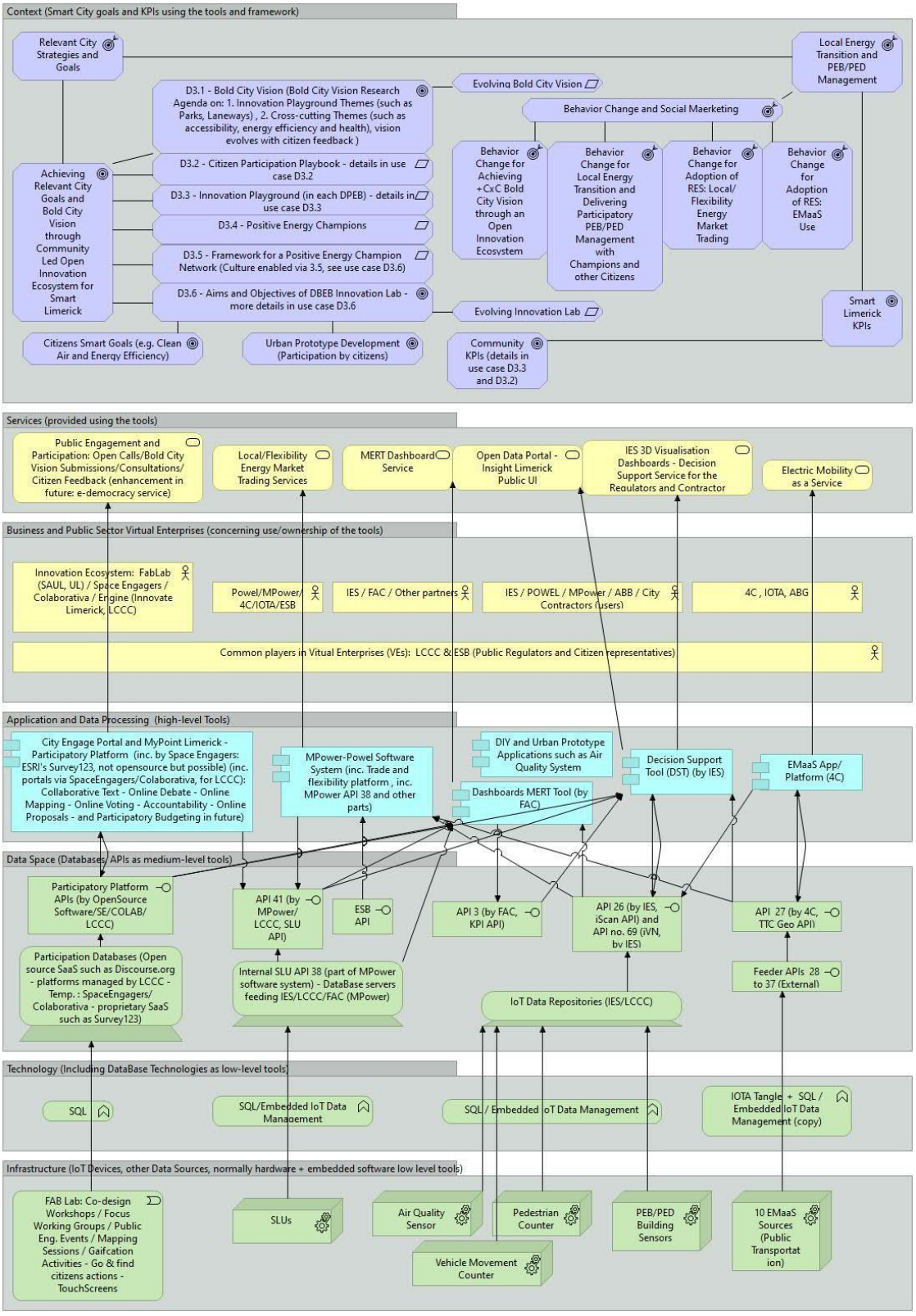


Figure - Several API entries in the API catalogue shown in the use case (D1.2)

## B.2 APIs for CommunityxChange Case for WP3

Based on the use case from WP3, CommunityxChange as discussed in D1.2 (section 6.3 CommunityxChange Use case). The APIs deployed in the CommunityxChange use case are shown in the table below.

| API number in catalogue | API name | API owner | API consumer | API description |
|---|---|---|---|---|
| 3 | GetKPI | FAC | LCCC | Getting KPI information from the MERT (+CxC dashboards, showing KPIs provided by different partners, used by LCCC and other stakeholders) |
| 26 | iSCAN (part of DST) | IES | LCCC | To allow configuration and use of a time series database for building monitoring and analysis. |
| 41 | Report and Partner API | MPower | LCCC | Provide information or anything (report) from MPower needs to provide to partners |

*Table: Description of the APIs in the CommunityxChange use case (implementation of WP3 frameworks for Limerick)*

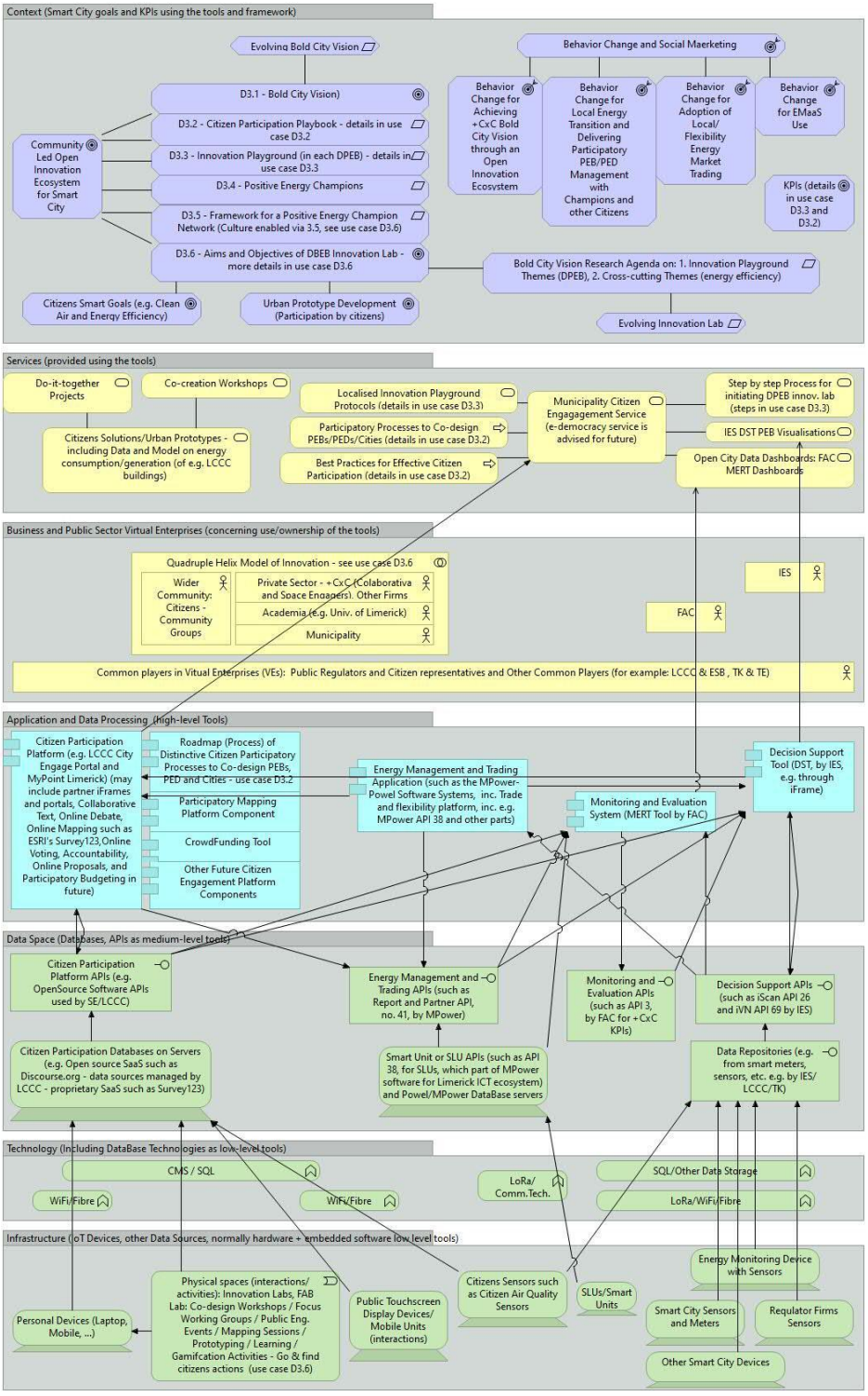


Figure - The relevant API entries in the API catalog are shown on the use case (D1.2)

API catalog entry no. 3 is, for example, used in the case (for more details of which please refer to Appendix G). A table containing the list of APIs with mentioning the owner firms and also an API description appears in Appendix G (more appears in the API Catalog itself).

In the subsequent subsections we discuss enhancements and best practices which should be used by partners to ensure data integration and interoperability. This involves present work for us to *enable* partners and future work for partners to *achieve* the final goal.

## B.3 APIs for the IOTA Module for Traded Flexibility Energy Marketplace

Based on the use case for the IOTA module for Traded Flexibility Energy Marketplace Use case as discussed in D1.2 (section A.10. The IOTA module for Traded Flexibility Energy Marketplace use case). The APIs deployed in this use case are shown in the next table.

| API number in catalogue | API name | API owner | API consumer | API(s) description |
|---|---|---|---|---|
| MarketplaceRegister API (53) | MarketplaceRegister | IOTA | Powel | Initial asset registration with Marketplace |
| Payment APIs (44, 45, 46, 47, 48, 49, and 50) | AssetInitiation API(44), AssetContract API(45), AssetPayment API(46), AssetPaymentsent API(47), AssetPaymentconfirmed API(48), AssetPaymentcancel API(49), and AssetPaymentclaim API(50) | IOTA | Powel | Enables a direct payment between the two involved assets, from consumer to producer |
| Settlement APIs (54, 55, 56, 57, 58, 59, 60, and 61) | MarketplaceOffer API(54), MarketplaceRequest API(55), MarketplaceMatchAPI(56), MarketplaceProvision API(57), MarketplacePaymentconfirmation API(58), MarketplacePaymentprocessing API(59), MarketplaceCancel API(60), and MarketplaceClaim API(61) | IOTA | Powel | Receive, Will move later to appendixsends, and shares data guaranting data integrity, immutability, and auditable within the marketplace |
| Data Logging APIs 51, 52, 62, 63, 64, 65, and 66 | AssetProduceenergy API(51), AssetConsumeenergy API(52), BidmanagerOffer API(62), BidmanagerRemove API(64), | IOTA | Powel | Verify the authenticity of shared data and uses agreed and |

| | | | | |
|---|---|---|---|---|
| | UserFund API(65), and UserNotifyevent API(66) | | | standardized data model for simple matching policy for bid management |

*Table: Description of the APIs in the IOTA Module for Traded Flexibility Energy Marketplace use case*

## B.4 APIs for the Trondheim Case

| API number in catalogue | API name | API owner | API description |
|---|---|---|---|
| 3 | GetKPI | FAC | Getting KPI information from the MERT (+CxC dashboards, showing KPIs provided by different partners, used by LCCC and other stakeholders) |
| 4 | eMaaS-locationlogging-location | IOTA | To store location data for a new or existing trip, the mobile app sends appId, tripId and location data parameters to the cloud backend /location API. |
| 5 | eMaaS-locationlogging-claim | IOTA | To retrieve location data in case of a claim for a selected trip, the mobile app sends appId and tripId parameters to the cloud backend /claim API. If the combination of appId and tripId is found, location data will be fetched from the Tangle, decrypted and sent to the mobile app. |
| 7 | ForecastAPI | TE | TE forecast to be provided to ABB via API |
| APIs no. 8 to 25 | IOTA's Transportation APIs Package | IOTA | Various APIs for recording/updating and retrieving transport providers, users, user journeys and payments, using IOTA Tangle (IOTA's Distributed Ledger Technology). |
| 26 | iSCAN (part of DST) | IES | Allows configuration and use of a time series database for building monitoring and analysis. IES |

| | | | |
|---|---|---|---|
| | | | Tool consisting of iVN (API no. 69) and iSCAN (API no. 26) provides building level and demand curve calculation, load curve and also grid calculation. iSCAN is a web-based tool from IES which is used to import, manage and analyze operational (time-series) data from various sources in a single platform. Measured building energy consumption data from smart meters, or simulated energy consumption data from the IESVE, can be uploaded to iSCAN. This data can then be imported into the iVN tool as a profile asset to be assigned to a building in the iVN model. |
| 27 | TTC Geo API | FourC AS | The TTC Geo API is an API designed to allow for the best possible integration with different map solutions. The API is mainly designed to output geographical data, which can be displayed and interacted on maps. The objects themselves are data rich, but it's up to each individual user of the API to utilize the information that is made available. The API is based on outputting the data as GeoJSON objects (RFC 7946). The GeoJSON specification lets the user easily take the data out as a standardized object format and display it on a map. GeoJSON is also very frequently supported natively by map solutions to display the objects. The API is based on GraphQL, a modern alternative to the more traditional REST. [...] GraphQL standard also supports subscription based requests over Websockets. The API is very focused in utilizing this concept, as the data is dynamic and changes over time[...] |
| 28 | Trondheim City Bikes | Trondheim City Bikes | Trondheim City Bikes - Feeds 4C (feeds partner API) |
| 29 | Nobil (charging | Nobil | Charging stations for EVs - Feeds 4C (feeds partner API) |

| | | | |
|---|---|---|---|
| | stations for EVs) | | |
| 30 | Entur (public transport, bus, train, boat, etc.) | Entur | Public transport, bus, train, boat, etc. - Feeds 4C (feeds partner API) |
| 31 | Rent-Centric | Rent-Centric | Rent-Centric - Feeds 4C (feeds partner API) |
| 32 | National Road Database | National Road Database | National Road Database - Feeds 4C (feeds partner API) |
| 33 | AVINOR (Norwegian Airports) | AVINOR | AVINOR (Norwegian Airports) - Feeds 4C (feeds partner API) |
| 34 | Adbshub.org (real time aircraft positions) | Adbshub.org | Real time aircraft positions - Feeds 4C (feeds partner API) |
| 35 | Trøndertaxi | Trøndertaxi | Trøndertaxi - Feeds 4C (feeds partner API) |
| 36 | AtB (for bus location) | AtB public transport authority | Bus location - Feeds 4C (feeds partner API) |
| 37 | Trondheim Airport Express Buses | Værnes-Ekspressen | Trondheim Airport Express Buses - Feeds 4C (feeds partner API) |
| 42 | MarketPlatformData | Powel | Allows sharing of time series data regarding energy forecast, plans, transactions, etc. |
| 43 | TK-KPIs | TK | Contains KPIs from Trondheim municipality, to feed the FAC's dashboard software system (API no. 3 is relevant) |

| | | | |
|---|---|---|---|
| APIs no. 44 to 66 | IOTA Marketplace Management API Package | IOTA | Marketplace bids, energy assets and payments management APIs through IOTA Tangle (IOTA's Distributed Ledger Technology).<br>May be used for double-checking and proving transactions. |
| 69 | iVN (part of DST) | IES | The main purpose of the iVN tool is to assist utility operators, policy makers and engineers with city or district level decision making with regards to the supply and management of natural and man-made resources and the devices that make that possible. Specifically, it can inform users of the current performance of various utilities and quantitatively predict the impact of changes to city-infrastructure. For example, adding a wind farm to a particular level of the electricity grid will generate electricity when it is windy, reducing the amount needed to be imported from abroad, reducing the carbon emission factors for the housing served by the wind farm and helping to populate storage devices with electricity in times of low demand; the effect on the whole city infrastructure is estimated naturally as a consequence of adding the wind farm to the network. iVN Communicates real time metered data with iSCAN, and communicates with other partner systems regarding PEB/PED models. |
| 71 | Charger and Battery Simulator API | FourC | The Charger and Battery Simulator (CBS) simulates a simple usage pattern of electric cars at electric chargers. The CBS was implemented as part of the +CityxChange project to enable other partners in the project to consume more or less realistic car usage of charging stations, the charging parameters and the car battery status, since no real data was available at the time. |
| 72 | ABB third party | Third party - | ABB third party Protocols and APIs supported |

| | Protocols and APIs | supported by ABB | |
|---|---|---|---|
| 73 | TTC API | FourC | o SIRI VM interface for vehicle position. Currently, only PT vehicles are available. Other modes of transport can be added when standardized in a newer SIRI/transmodel version.<br>o NeTEx - route network data. Currently, supported for PT routes only.<br>o Bus stop position - TTC offers an export of bus stops, using concepts like Quay and StopPlace, as defined by Transmodel. Currently supports all stops defined in NVDB.<br>o Charger data - Positions and status/attributes, historical and real-time<br>o EV data - Core data like positions and status/attributes, historical and real-time<br>o Vehicle availability<br>o Journey description<br>o Road closures and status<br>o All data are stored historically. This means that the current status of an object and its attributes for a specific moment back in time can also be queried for. |

*Table: Description of the APIs for the Trondheim Case*

## B.5 Overview of the APIs connected to the EA Use Cases, aligned with D1.2

Following is the table showing *APIs* from the API Catalog against the *use cases* described in D1.2 (Petersen et al., 2021) followed by the section in D1.2 in which the use case appears.

| Use Case | API Numbers Relevant from the API Catalog |
|---|---|
| Limerick Case (D1.2 Sec. 6.5) | 3, 26, 38, 41, APIs no. 44 to 66 (package) and 69 |
| Trondheim Case (D1.2 Sec. 6.4) | 3, 4, 5, 7, APIs no. 8 to 25 (package), 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 42, 43, APIs no. 44 to 66 (package), 69, 71, 72 and 73 |
| CommunityxChange Case (D1.2 Sec. 6.3) | 3, 26 and 41 |
| Common Energy Market Case (D1.2 Sec. 6.2) | 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, and 66<br>APIs no. 44 to 66 (package) |
| Integrated Planning and Design Case (D1.2 Sec. 6.1) | 26 |
| Seamless eMobility System Including User Interface Case (D1.2 section A.1) | 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37 and 44<br>APIs no. 8 to 25 (package) |
| IOTA eMaaS Payments Trail and Tech Specs and Requirements Case (D1.2 sec. A.2) | APIs no. 44 to 66 (package) |
| Limerick DST (Integrated Modelling and Decision Support Tool) Case (D1.2 sec. A.3) | 3, 26, 38, 41 |
| Delivery of the Citizen Participation Playbook Case (D1.2 sec. A.4) | 67 |
| Framework for an Innovation Playground Case (D1.2 sec. A.5) | 26, 69 |
| MPower Tool Case (D1.2 sec. A.6) | 38, 39, 40, 41 |
| Powel Tool – (Grid Operation Tool) Design and Operation of Local Energy System Case (D1.2 sec. A.7) | 2, 3, 5, 7, 8, 26, 42, 69 |

| | |
|---|---|
| The Common Energy Market Case (D1.2 sec. A.8) | 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, and 66 APIs no. 44 to 66 (package) |
| Community-led Open Innovation (Framework for DPEB Innovation Labs) Case (D1.2 sec. A.9) | 69 |
| The IOTA module for Traded Flexibility Energy Marketplace Case (D1.2 sec. A.10) | 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, and 66 APIs no. 44 to 66 (package) |
| Local DPEB Trading Market Demonstration Tool Case (D1.2 sec. A.11) | 7, 42, 72, APIs no. 44 to 66 (package) |
| Continuous Intraday Trading: Integration Between Project Partners Case (D1.2 sec. A.12) | 7, 42, 72, APIs no. 44 to 66 (package) |
| Optimization Model (Microgrid Balancing and Optimization) Case (D1.2 sec. A.13) | 7, 42 |
| (Integrated) Booking System for Space Case (D1.2 sec. A.14) | – |
| Citizen Power Platform (DECIDIM (Playable Trondheim) Case (D1.2 sec. A.15) | – |
| Integrated Investment Model (D1.2 sec. A.16) | 26, 42, 45, 69 |

# Appendix C – +CityxChange API Catalog and Repository Web Portal

An online API catalog was developed that contains the API specifications provided by the partners. The web app is implemented in PHP/MySQL (back-end) and HTML-JavaScript (front-end) on a WAMP server[27].

We have considered the most important meta-data of APIs in the catalog fields while the API specification field should contain the full specifications of the API including all of its meta-data and value data schemas (see the example in the following figure). Briefed through Data Meetings, partners register, login and add API info. A 3rd party developer or partner of the smart city can easily access this portal and search for specific APIs to obtain the complete specification for development purposes (e.g., URL, API name, Type of API, description, etc.). The catalog content is the responsibility of the ICT partners. Some APIs are still being developed/documented by the partners at the time of writing this report.

The basic functionality includes adding and editing APIs (see example below); removing APIs; browsing and searching APIs; export of CSV reports; structure aligned with the API questionnaire format as agreed with the partners. The catalog website is accessible through a login.

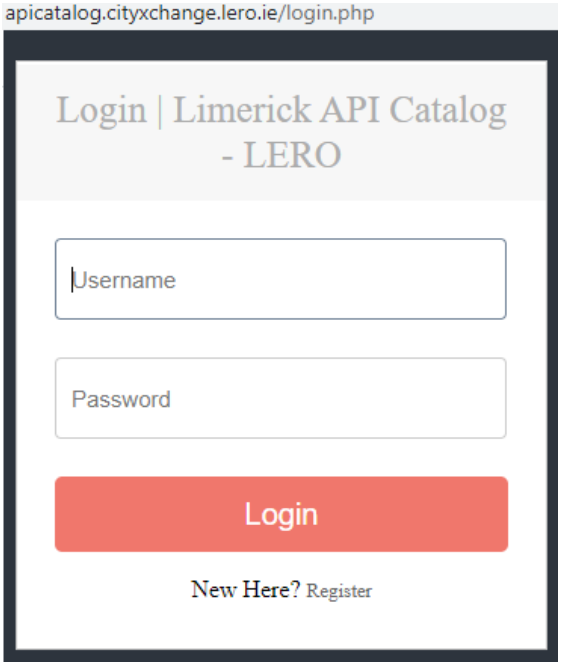


Figure: The API catalog is hosted at Lero, University of Limerick for +CityxChange

[27] http://www.wampserver.com/en/

## API Catalog - CityXchange - Limerick : Add new API information

| | |
|---|---|
| api_owner | |
| contact_person | |
| contact_email | |
| api_name | |
| url | |
| api_objective | |
| api_version | |
| protocol | ◉ REST<br>○ SOAP<br>○ Web Socket |
| data_format | ◉ JSON<br>○ XML |
| open_api | ◉ Yes<br>○ No |

Figure: API property view in the API catalog

# +CxC API Documentation

## HOST: https://Limerick-api.app.ie

API Public Id: 1

KPI Public Id: 14

API Type: REST

Application Protocol: HTTP

Response Data Format: JSON

Frequency of Reporting: Monthly

Reporting Date: 6

Unit of Measurement: kWh/msquare

Access Type: Public

Access Time Starts At: None

Access Time Ends At: None

Entity Type: KPI

.
.
.

# Proposed FAC API structure

**Retrieve KPI data (Option 1) [GET]**

**Parameters:**

- KPI_ID: 14 (int)
- City Name: 'Limerick' (string)

**Response 200**

*Model (application/json)*


```
{
  "meta": {
    "Location": "Limerick",
    "Unit of Measurement": "kWh/msquare",
    "Aggregation Data Resources": "SLU, Building Information",
    "Calculation Period": "Month"
  },
  "data": [
    {
      "month": "1",
      "KPI Value": 62
    },
    {
      "month": "2",
      "KPI Value": 51
    }
  ]
}
```


Figure: An example of an API specification as stored in the API catalog (from API no. 3, also compare specification example given in Annex E)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 36 | AtB public transport authority | [redacted] | [redacted] | AtB (for bus location) | http://TBD | TBD | Bus location | TBD |
| 37 | Værnes-Ekspressen | [redacted] | [redacted] | Trondheim Airport Express Buses | http://TBD | TBD | Trondheim Airport Express Buses | TBD |
| 38 | MPOWER | [redacted] | [redacted] | SLU API | https://smartnode.net/node | TBD | Supply raw voltage, current and power factor data from prosumer SLUs | 1 |
| 39 | MPOWER | [redacted] | [redacted] | Operator API | http://TBD | TBD | internal use by MPower employees | TBD |
| 40 | MPOWER | [redacted] | [redacted] | Prosumer API | http://TBD | TBD | used by prosumers for trading, billing control, home management | TBD |
| 41 | MPOWER | [redacted] | [redacted] | Report and Partner API | http://TBD | TBD | anything (report) MPower needs to provide to partners | TBD |
| 42 | Powel | [redacted] | [redacted] | MarketPlatformData | http://TBD | TBD | Market Platform Data | TBD |
| 43 | TK | [redacted] | [redacted] | TK-KPIs | https://tipqa.trondheim.kommune.no/cxc/v1/kpi/ | No | Providing TK KPIs | 1.0.0 |
| 44 | IOTA | [redacted] | [redacted] | AssetInitiation | https://{asset_URL}/init | No sharing of personal data | Remote Initialization of IOTA asset module | 1.0.0 |

Figure: Example view of APIs in the API Catalog

A concise API Table with more details appears in Appendix G, the APIs are used in D1.2 use cases with the same number as their number in the table/catalog, connecting the API definitions to the demo projects and the +CityxChange ICT ecosystem diagrams.

## Appendix D – API Specification Languages and Tools

We introduce six major existing API specification standards/tools to define and describe an API. To define an API we can use one of the following languages: JSON, YAML, Markdown, RAML or WADL. An API specification is always linked to a data-model that defines the entities' names, types, links, and constraints. These entities are important as names of attributes in the data structure of each method of an API. In this subsection, we mention a number of API specification standard languages with their supportive tools and we illustrate more important ones by examples showing the different aspects. A number of API specification languages comparison criteria will be used in the next section for the comparison of API specification languages to show the reason behind choosing one language (API Blueprint, which we advise partners to use, or the OpenAPI). Overall, the actual specification used is a minor aspect and the actual specification language used should be an internal issue of the partners and their development process, which should not strongly influence the API exchange.

### 4.1.1 OpenAPI specification - Swagger

Swagger is an API framework specification based on JSON (and supports YAML starting from version 3) and is designed to be both machine and human readable[28]. It is mainly used to create RESTful interfaces that map resources and operations assigned to an API. This allows the API to be discovered by external tools/users without the need to access its source code or any other documents. Swagger editors are provided with code generators that allow to generate server implementations directly from the specification of the API. For some partners, OpenAPI is the preferred version and is supported as a solid option. This makes it easy to develop an API with early contact testing for rapid prototyping.

### 4.1.2 RAML

RAML is a modeling language to support RESTful APIs. RAML is supported by different IDEs. The most known one is API workbench[29] that is developed for ATOM[30]. It is a powerful language, especially using the API workbench and API designer tools.

### 4.1.3 API Blueprint (chosen)

API Blueprint[31] is a formal and simple structured language to specify API. A table containing comparison criteria showing why we have chosen it follows in section 4.1.5. API blueprint relies on the MarkDown[32] language that uses a limited document syntax. API blueprint is easy for all actors that are involved in the API lifecycle. With API Blueprint one can specify any type of APIs (e.g., SOAP, REST, Websockets, etc.). It is supported by a large number of tools[33] for editing, testing, parsing, rendering, converting and lexering. It is also supported by Mock-servers in order to generate a simple server (hosting web services) starting from an API Blueprint specification document.

**API Blueprint overall format in brief:** In order to simplify the API specification through API Blueprint, we mention the basic and most important guidelines to structure the API Blueprint document.

1. The meta-data that should be considered during the API implementation are mentioned at the beginning of the API Blueprint document. They should be declared in a name:value format.
2. A general description of the API (indicated by # at the beginning of the line to refer to the title as a header) including: objective of the API, main characteristics, expected

[28] https://swagger.io/specification/
[29] https://atom.io/packages/api-workbench, https://github.com/mulesoft/api-designer
[30] https://atom.io/
[31] https://apiblueprint.org/
[32] https://www.markdownguide.org/
[33] https://apiblueprint.org/tools.html

outputs and input parameters, and the instructions that can help developers understand and implement the API clearly and easily. This description can be divided into different sections.

3. The different main methods that identify the functionality of the API. For example, in the case of HTTP applications, the API is required to return responses relying on HTTP protocol specifications (methods such as GET, POST, PUT, DELETE), and the response should indicate the corresponding HTTP status code (e.g., 200 for OK)

Every method that exchanges data with the requester has inputs (called parameters) and output. The output should identify the data format (e.g., JSON) and structure that is going to be sent. These information are added in the Response subsection. The structure of the data is an array in the Model subsection that is branched from the Response keyword of every method. This model is declared using: Model (application/[data format]). It is recommended to provide the response data structured in meta and value data.

API Blueprint Example: Thanks to its simplicity and clarity, we used as an example (and we generally advise) API Blueprint to create an API specification. In the following example, the KPI is relevant to FAC (Monitoring & Evaluation system) and after mutual discussions in a meeting we agreed on the following initial specification. The example considers the KPI number 14 (RES Efficiency) from D7.1 and can be used by partners as an example.

Please refer to the following Appendix E for a detailed code example.

RES Efficiency API Blueprint specification:
This API provides access to the RES efficiency KPI. The RES Efficiency stands for Renewable Energy Source Efficiency. It is defined as, kWh/sqm (UFA) per year improved energy efficiency (final energy demand). An approach in the creation of DPEBs by including energy efficiency measures that will reduce the overall energy demand of the building (as part of the larger block). Proposed solutions are the improvement of building airtightness levels, mechanical heat recovery, demand controlled ventilation, reduced reliance on fossil fuel energy systems, use of RES integrated into buildings with the ability to share energy through the building and home energy management systems.

The presented API specification in Appendix E defines a RESTFul API that uses JSON format to exchange data between partners. We have agreed that JSON provides better data throughput performance than XML. Therefore, we have defined the application as using JSON with input parameters and a model for the output. This model is structured using JSON format and we divided the output model into two main parts: {value and meta} data. From its definition, the meta data defines the identifiers and data sources (simply it defines how to read and use the data), while the value data contains the values in numbers which is

the objective of the request. The data in that structure is defined using key-value declarations {"key":"value"}. The key refers to the name of the attribute, while the value stands for the value assigned to that attribute.
The developer/programmer can understand this specification easily and use meta-data at the beginning as variables and consider them during the implementation of the API. On the other hand, the structure of the data model(s) can be easily iterated over to look for the value per each key (attribute). Moreover, we have recommended the developers (through the general description section of the API specification document) to use regular expressions including all possible cases to define the possible routes (http routes) in the controller implementation when using Model-View-Controller (MVC) approach. Examples of MVC frameworks that can be used to implement such an API are: Symfony/PHP with Doctrine (ORM), React native/NodeJs with Meteor (ORM) or Mongoose (ODM), ASP.NET with Microsoft Entity Framework (ORM), Javascript or GOLang with Prisma framework (ORM-ODM), and Java/J2EE with Spring and Hibernate framework as an ORM.
Relying on API Blueprint enables the different partners to generate Javascript codes for the client side, and generate web service code (server side) automatically as well, using support tools[34]. This mechanism facilitates and hides the complexity of software development and enables the software developers to improve the code as well in correspondence to the requirements mentioned in the API specification document (specially regarding meta-data at the beginning of the document). This specification is going to be modified in different versions according to the updates and meetings with different partners and to the new design concepts that we are going to define.
In the API catalog, the API specification can be added as markdown text as one of the fields. This feature facilitates the development of APIs in both current and future stages after the end of the project. API specifications include: meta-data, descriptions, requests and responses, etc.

### 4.1.4 Rapid-ML

Rapid-ML relies on realization modelling (http://rapid-api.org/rapid-ml) where a data model is shared between all APIs and each API uses some of the entities according to its needs. The data model defines the types that are used in each API thus it defines their business domain. This approach focuses on building the data model as a starting and shared point for designing the API. Rapid-ML is supported by a powerful IDE (RepreZen- API studio) that can be used for Swagger (OpenAPI) as well. This IDE provides means for the developers to exclude properties of an entity in the original data model and take what is needed through defining a set of specifications per each API according to its needs. There are two limitations for this approach: Complexity according to the necessity of a data model; RepreZen-API studio is not open source software.

[34] https://apiblueprint.org/tools.html

### 4.1.5 API Blueprint Choice: API specification languages comparison

Following is a comparison table with the API specification languages *criteria* we use:

| API Specification / Criteria | Machine readable | Simplicity & Clarity | Development (IDEs) | Supported API types |
|---|---|---|---|---|
| OpenAPI (Swagger) | ++ | + | ++ (Swagger repository) | + RESTful only |
| RAML | ++ | + | ++ (API workbench) | + RESTful only |
| **API blueprint** | **++** | **++ (Markdown)** | **++ (many)** | **++** |
| Rapid-ML | + | + | ++ (RepreZen- API studio) | ++ |

Both API blueprint and RAML are well supported by open development platforms. API blueprint supports all API types while RAML is only for RESTful ones.
Rapid-ML raises the importance of considering an important role of the data models shared between the different partners as a formal structure. This structure should consider the different aspects of data integration and privacy (e.g., GDPR rules).

**Conclusion:** From the recent analysis in the previous section, we decide to use API Blueprint before RML according to its wide range of usage regarding all types of APIs. Moreover, it is more clearer thanks to the markdown language used to specify the API. This makes it more documentable. A smart city has many partners who have already existing APIs that were developed before. It will be complex and expensive for them to respecify/re-implement APIs that may not be necessarily REST APIs. API blueprint satisfies our requirements and provides an acceptable solution for our project
Some partners have mentioned the need to prioritise ease of effective use by developers, and that (internal) specifications are not the main considerations. In some cases, Open API (Swagger) is considered a better choice.

# Appendix E – API Blueprint Specification Sample (KPI collection)

An API specification example in Blueprint referred to in the previous appendix, for the MERT API (collection of consortium KPIs). The example is filled in with sample reporting data on an energy KPI for Limerick. Further details are discussed in Section 4.

```
# +CxC API Documentation
## HOST: https://Limerick-api.app.ie
API Public Id: 1
KPI Public Id: 14
API Type: REST
Application Protocol: HTTP
Response Data Format: JSON
Frequency of Reporting: Monthly
Reporting Date: 6
Unit of Measurement: kWh/square , kWh/sqm
Access Type: Public
Access Time Starts At: None
Access Time Ends At: None
Entity Type: KPI
Result of Aggregation: True
Data Sources: {SLU, Building Information}
Smart City: Limerick
KPI Owner: {Partner Name +CityxChange}
Data Owner: {Partner Name +CityxChange}
Data Aggregator: {Partner Name +CityxChange}
Data Provider: {Partner Name +CityxChange}
Contract: {to be defined}
Data Collection Level: Building
Data Aggregation Level: Sub-City District
Inter City Exchange Allow: True
Intra City Exchange Allow: True
# RES Efficiency API


    # Data Requirements of the KPI before calculation
    Primary energy demand, Thermal energy demand, Primary energy factor for thermal
    energy, Electrical energy demand, Primary energy factor for electrical energy,
    Usable floor area of buildings, Number of buildings with EE interventions
    implemented.

        **Retrieve KPI data** **[GET]**

        **Parameters:**
        - KPI Name: 'RES Efficiency' (string)
        - City Name: 'Limerick' (string)
        - Month: '2019-07-01' (string)

        **Response 200**
        *Model* *(application/json)*
        ```
        {
         "data":{
         "KPI Value": 62,
         "Value Unit": "kWh/msquare",
         "Calculation Period": "Month"
         },
         "meta":{
         "code": 200,
         "Location": "Limerick",
```


```

```
 "Aggregation Data Resources": "SLU, Building Information"
 }
}
```
**DELETE KPI data** **[DELETE]**
Delete method is not allowed

- Response 405

**Update KPI data** **[PUT]**
Put method is not allowed

- Response 405

*Should be enabled in later stages of the project to update data. Functionality availability based on user role*

**Create KPI data** **[POST]**
Post method is not allowed

- Response 405

*Should be enabled in later stages of the project to add new data. Functionality availability based on user role.*

# Proposed FAC API structure

**Retrieve KPI data** **(Option** **1)** **[GET]**

**Parameters:**
- KPI_ID: 14 (int)
- City Name: 'Limerick' (string)

**Response 200**

*Model* *(application/json)*

```
{
 "meta": {
 "Location": "Limerick",
 "Unit of Measurement": "kWh/msquare",
 "Aggregation Data Resources": "SLU, Building Information",
 "Calculation Period": "Month"
 },
 "data": [
 {
  "month": "1",
  "KPI Value": 62
 },
 {
  "month": "2",
  "KPI Value": 51
 }
 ]
}
```
**Retrieve KPI data** **(Option** **2)** **[GET]**
**Parameters:**
- City Name: 'Limerick' (string)

**Response 200**

*Model* *(application/json)*

```


```

```
[
 {
 "meta": {
  "kpi_id": "14",
  "location": "Limerick",
  "unit_of_measurement": "kWh/msquare",
  "aggregation_data_resources": {
  "level": "Sub-city district",
  "...": ""
  },
  "calculation_period": "Month"
 },
 "data": [
  {
  "identity": "bldg1",
  "month": "1",
  "title": "",
  "value": 62
  },
  {
  "identity": "bldg2",
  "month": "1",
  "KPI Value": 31
  },
  {
  "identity": "bldg1",
  "month": "2",
  "KPI Value": 42
  },
  {
  "identity": "bldg2",
  "month": "2",
  "KPI Value": 51
  }
 ]
 },
 {
 "meta": {
  "kpi_id": "15",
  "location": "Limerick",
  "unit_of_measurement": "kWh/msquare",
  "aggregation_data_resources": [
  {
   "type": "level",
   "value": "City Level"
  },
  {
   "...": "..."
  }
  ],
  "calculation_period": "Month"
 },
 "data": [
  {
  "identity": "bldg1",
  "date": "MMM-yyyy",
  "KPI Value": 62
  },
  {
  "identity": "bldg2",
  "date": "MMM-yyyy",
  "KPI Value": 51
  }
 ]
 }
]
```


For the data model used in the above, FIWARE Data Model has also been considered (but not fully used). Relevant FIWARE Data Model for Key Performance Indicators appear in: (FIWARE Contributors, 2016)).

# Appendix F – Wider Range of Related Standards, Resources, Projects

This annex compiles a wide range of standards and resources that have been examined and discussed. They are listed here as further reading.

## F.1 EU Open Data Portal

A central web portal has been created for European Union to share open data standards and resources, including vocabularies. Categorisation of the data sets by subject helps with accessing/studying the datasets for reuse or metadata production/replication, while use of common vocabularies and standards for metadata is encouraged. Some parts are quite specialised to a specific task/study. Generalisation and development of resources to enable data integration and interoperability is ongoing. One can search for the metadata in the EU Open Data Portal via a SPARQL endpoint query editor:

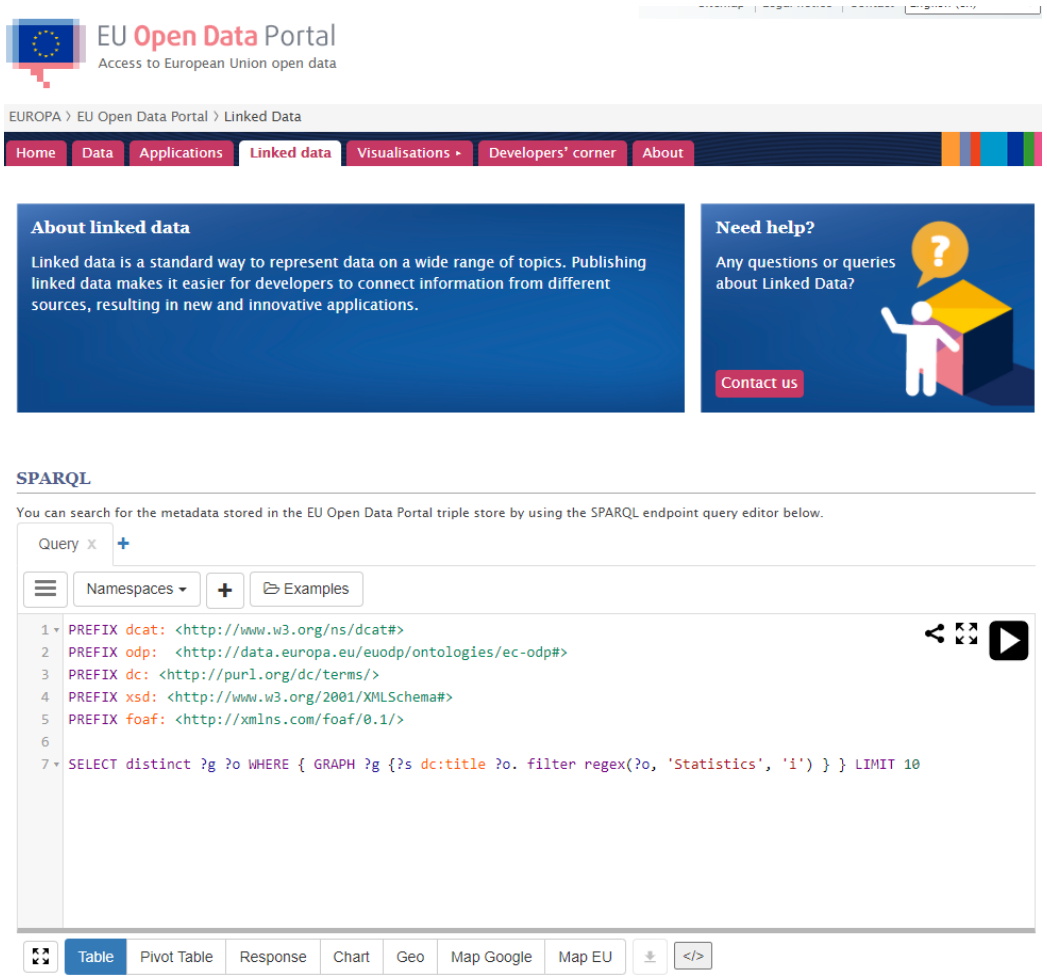


Figure - EU Open Data Portal *metadata search* facility[35]

## F.2 EuroVoc (EU Vocabulary)

European Union has a central place for EU Vocabularies. It's a thesaurus covering the activities of EU countries especially in the European Parliament. Some other vocabularies from other EU institutions and bodies are covered in EuroVoc. It seems to be an ongoing program many parts of which are yet to be developed and enhanced (including additions such as the European Legislation Identifier (ELI) ontology). This is an underdevelopment yet important work which can be benefitted from for data integration.

## F.3 DCAT Application profile for data portals in Europe (DCAT-AP)

Part of EuroVoc, DCAT-AP is a Public Sector metadata specification standard based on the Data Catalogue vocabulary (DCAT). There are implementation guidelines for DCAT-AP.

[35] https://data.europa.eu/data/datasets

DCAT-AP is officially aimed at data integration and interoperability (inclined towards public sector datasets):

1. Helps data reusers find and use the datasets developed by others.
2. Helps data providers with achieving wider reuse of their data by making it searchable and more accessible.

## F.4 EU Core Vocabularies

Introduction of widely accepted/supported open vocabularies help with *data integration*. While semantic processing for data integration can be costly, with inaccuracies, accepting common data models and vocabularies - which have been studied and standardised to be benefited from widely - is a good pathway for Smart Cities. EuroVoc is one of such resources.

According to EuroVoc, Core Vocabularies are "simplified, reusable and extensible data models that capture the fundamental characteristics of an entity in a context-neutral fashion". Following public sector core vocabularies are covered in EuroVoc:

1. The Core Business Vocabulary (latest version).
2. The Core Location Vocabulary (latest version).
3. The Core Person Vocabulary (latest version).
4. The Core Public Organisation Vocabulary (latest version).
5. The Core Public Service Vocabulary (latest version).
6. The Core Evidence and Criterion Vocabulary (latest version).

All in all, public sector vocabularies and other data integration and interoperability resources cover topics most of which are also important for the private sector because of shared concepts between sectors or because of the need for regulatory compliance.

## F.5 FIWARE

FIWARE[36] is and possible future pathway for development of the +CityxChange and other smart city consortiums' efforts. It provides an open source platform, including open data model (as described earlier, see Section 3), framework and tools (such as API Management tool) for smart city ICT ecosystem development.

FIWARE Catalog (FIWARE Foundation, 2021) can be an important element of Smart City development programs in Europe, including for +CityxChange.

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

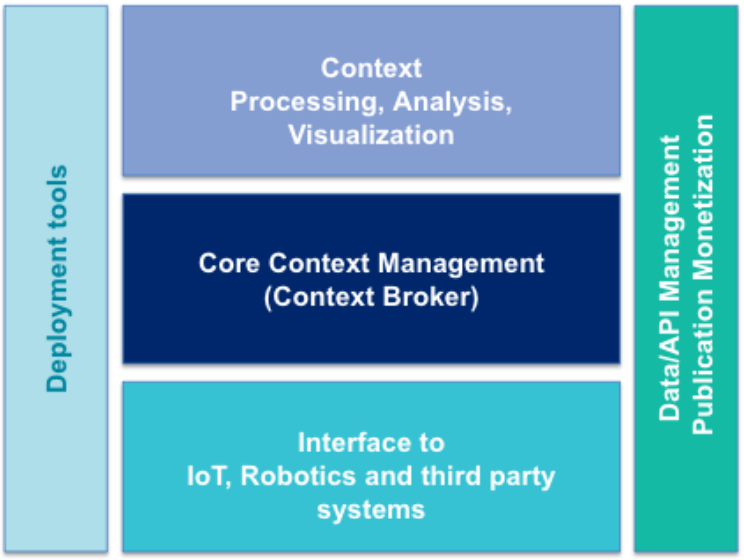


Figure - FIWARE Catalog Diagram (FIWARE Foundation, 2021)

## F.6 Joinup

Smart City projects can involve many partners (30 in case of +CityxChange) each with varying needs which may fall out of the developed frameworks/solutions at hand, in which case the solution should be built in a specialised way after consulting relevant networks and communities (to avoid reinventing the wheel).
Joinup is a central web including a network of 15k+ people (Sep. 2020), created by The European Commission to enable the public sector, businesses and citizens to share/reuse solutions and best practices related to ICT and data (interoperability).

## F.7 ADMS

Asset Description Metadata Schema Application Profile (ADMS-AP) specification, at Joinup, using which publishers of interoperability solutions "describe their content and federate it".

## F.8 ISA2

The important European interoperability network. providing "Interoperability solutions for public administrations, businesses and citizens". The data integration and interoperability framework we are developing (reported from section 3 onwards) could be eventually contributed to ISA2 network. Many of the solutions and frameworks here (and on Joinup, in general) tend to be quite specialised and cannot be used in other (even neighbouring) contexts without an *extension and customisation* project. Although our data integration and interoperability framework is developed for +CityxChange, it will be among the relatively-more-general ones in ISA2, from which other smart city projects can benefit more quickly.

## F.9 ISO/IEC 30182:2017

This is an ISO standard for "Smart city concept model — Guidance for establishing a model for data interoperability"[37], and covers semantic interoperability: Aimed at organizations that provide data-intensive services to communities in cities, as well as city regulators, it provides guidance on a smart city concept model (SCCM) to provide the basis of interoperability between component systems of a smart city, by mapping from a local model to a parent model and aligning the ontologies in use across different sectors. It includes concepts (e.g. ORGANIZATION, PLACE, COMMUNITY, ITEM, METRIC, SERVICE, RESOURCE) and relationships between concepts (e.g. ORGANIZATION has RESOURCEs, EVENT at a PLACE). It does not cover the data standards that are relevant to each concept. The SCCM has been devised to communicate the meaning of data. It does not attempt to provide concepts to describe the metadata of a dataset, for example, validity of data.

## F.10 International Telecommunication Union (ITU)

There is a portfolio of standards in the context of smart cities (UNE 17804:2015) "Telecommunication standards". The reference architecture proposes an interoperability layer on top of the knowledge layer. The knowledge layer represents the data analysis relying on the city semantics that include the meta-data repositories. The REST APIs are considered as a support software in the support layer (aligned vertically with all layers). These APIs provide data share and publish through open data/API platforms. Collection technologies and systems are positioned at the bottom of the proposed architecture (it contains social networks, sensor networks, etc.). The communication between collection system and knowledge layer components is done through REST, WebSockets, ... applications.
ITU also has a DLT standard helping with robust and secure data exchange between systems (used in section 6).

## F.11 INSPIRE

INSPIRE[38] or Infrastructure for Spatial Information in Europe (Tóth & Smits, 2007) which can be used by partners who need standardised reference to spatial information and can be a point of agreement for data integration surrounding this aspect of the smart city data, using a common and standard data model. INSPIRE itself is a large-scale data federation, which may provide some underlying spatial data that cities are using. Directly connecting the project is not feasible, while some underlying standards may be informative[39].

[37] https://www.iso.org/standard/53302.html
[38] https://inspire-geoportal.ec.europa.eu/
[39] https://inspire.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_BU_v3.0rc3.pdf

## F.12 Transmodel

Transmodel[40] is a reference data model for public transport for Europe which can be used for EmaaS (Electric Mobility as a Service) and other transport related data in +CityxChange. As the base model is old (dating back to 90s), necessity of updates may be examined by partners and an improved model sought/done such that it is properly usable. The problems regarding maintenance and enhancement of Transmodel, together with solution overview, is discussed[41]. We have noticed that partners in +CityxChange do not usually fully use a ready data model and consider and usually do modifications specific to the task at hand. This can be addressed by Semantic Interoperability in future, or agreement on further developed data models in future which take care of many sub-situations (one of which might match the partner problem closely enough to merit full compliance. this might be gradually happening regarding FIWARE and some other active resources we have mentioned in the report).
Transmodel is used in the eMaaS cases, as described in Section 5.

## F.13 BSI PAS 182 and BSI PAS 212

PAS 182[42] by BSI is a concept model for smart city, which has a guide for establishing a data model and can help with data integration and interoperability specially for those firms working in the UK, while other firms can also benefit from it. According to BSI, PAS uses the public sector concept model (PSCM), a basis for interoperability at the upper ontology level, while outlining details of the smart city concept model (SCCM). More information on PAS 182 and how to use it can be gained using the link in the footnote. BSI PAS 212 is a corresponding API specification. It is about automatic resource discovery, including APIs, for IoT, which helps with IoT interoperability and with openly sharing resources in an integrated discovery system[43]. Most of +CityxChange partners may prefer Europe-based resources accessible through ISA2/Joinup network and FIWARE community.

## F.14 READY4SmartCities

It was an FP7 project trying to provide roadmap and interoperability for energy systems of smart city[44]. It provides ontologies and datasets for various aspects of smart city energy. Energy partners usually use such resources to gain insight and won't rely with their systems

[40] http://www.transmodel-cen.eu/transmodel-at-a-glance/
[41] Transmodel updates: http://www.transmodel-cen.eu/ also refer to http://www.transmodel-cen.eu/transmodel-at-a-glance/
[42] BSI PAS 182: https://www.bsigroup.com/en-GB/smart-cities/Smart-Cities-Standards-and-Publication/PAS-182-smart-cities-data-concept-model/
[43] BSI PAS 212: https://www.bsigroup.com/en-GB/about-bsi/media-centre/press-releases/2016/july/Internet-of-Things-interoperability-specification-is-published/
[44] FP7 READY4SmartCities project: https://cordis.europa.eu/project/id/608711

fully on the resource. This is both due to the specific cases they have at hand and also due to the already developed systems. Another example of such a pattern is FAC's use of FIWARE data model.

## F.15 oneM2M

A global open technical standard for interoperability developed from early 2010s by eight firms from different countries (from Japan to Europe to the US). Their interoperability enabler may be of particular interest[45]. The IoT, smart grid, connected car and home automation aspects of this standard are bold and their usage can be further examined[46], where partners face exceptions not covered by the DIIF *inner circle* of standards and best practices. We do not advise it as one of the *inner circle* standards for DIIF, as our interoperability recommendations are mainly built surrounding federation of APIs, while oneM2M is developing a unified interface for abstraction and around semantics of specific technical data for various devices and systems.

## F.16 Dublin Core

Well-known resource (document and media) metadata definition initiative[47]. Metadata design, implementation and best practice can help with data integration surrounding common standards, or make definitions more aligned and more ready for semantic interoperability efforts (including automatic seamless integration of slightly deviating data models) in future.

## F.17 W3C SSN and Other Resources

There are many standards and resources which may be of less relevance given the present definition and scope of +CityxChange, such as W3C SSN (Semantic Sensor Technology)[48]. When the sensor network in the smart city is developed further, such a relevant ontology may help. At the time being, representing sensor data is on the shoulder of in-house definitions and data models by partners such as MPower, for the very specific tasks they have at hand, and the data is communicated using APIs such as, for example, MPower's SLU API (API no. 38 in Appendix G and the API Catalog).

[45] https://onem2m.org/images/files/oneM2M-whitepaper-January-2015.pdf
[46] http://onem2m.org/technical/published-drafts
[47] User guide: https://www.dublincore.org/specifications/dublin-core/usageguide/1998-05-23/
Definition: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/
[48] https://www.w3.org/TR/vocab-ssn/

## F.18 OASC Minimal Interoperability Mechanisms (MIMs)

Open Agile Smart Cities (OASC)[49] is a network of cities and city support. As part of its work, it has suggested MIMs to help with achieving interoperability between smart cities and their suppliers. Referring to their MIM wiki[50] is advised to more-interoperability-active partners. It helps with interoperability through common data models, context Information Management API and Marketplace API. TK and LCCC are members.

## F.19 Transport.API[51]

It is a service that provides aggregated open data. It provides (via REST API) a relevant number of datasets integrating both static and real-time data for mobility aspects. It is the basis for one fourth of the relevant transportation Apps in the UK in early 2021, making it number one for the British market.

## F.20 ECIM[52]

A platform for intelligent mobility to integrate the data collections and unify the format by converting it to JSON and then storing them in a common ECIM MySQL database.

## F.21 Km4City[53]

An open urban platform which provides an API model and data aggregation related to the mobility and transport in smart cities.

## F.22 TM Forum[54]

It has resources for open API development and use including a developer portal, open digital framework and open digital architecture.

## F.23 CitySDK

This is a service development kit for cities and developers that aims to harmonize APIs across cities[55]. It provides a way to link between the datasets that are coming from the different vendors (datasets have no initial relations/links between each other).

[49] https://oascities.org/
[50] https://oasc.atlassian.net/wiki/spaces/OASCMIM/overview
[51] https://www.transportapi.com/
[52] https://www.cloudwatchhub.eu/serviceoffers/ecim-european-cloud-marketplace-intelligent-mobility
[53] https://www.km4city.org/
[54] https://www.tmforum.org/open-apis/
[55] Smart City Service Development Kit and its Application Pilots (CitySDK), https://cordis.europa.eu/project/id/297220

## F.24 EIP-SCC Urban Platform

One of the EIP-SCC (European Innovation Partnership – Smart Cities & Communities) action clusters on Urban Platforms has compiled overall requirements[56], including those for Interoperability:

1. Cater for interoperability in urban infrastructures
2. Enable replicability of solutions city-to-city
3. scale without technical constraints & cost increase
4. provide open standardised APIs & SDKs
5. Enable real-time capabilities
6. Support implementation of functional and technical capabilities.

Its reference architecture classifies the 4th layer (Integration and Orchestration Capabilities) as having the following components: Data Exchange, Messaging, Load Balancing, (Open) API Management, Rules Management, Event Management, Transaction Management, Process Orchestration & Monitoring, Service Management, Publish – Subscribe & Notification Management, Collaborate Communication & (Social) Media, Personalization, Ecosystem MarketPlace. Its 3rd layer which is the Data Management & Analytics Capabilities includes the following components: Data Ingest, Data Virtualization, Non-time series Data integration & transfer, Time series Data integration & transfer, Data Fusion, Data Aggregation, Complex-event processing, Data Logistics, Data Privacy protection, Data Security Management, Data Assurance Management, Data Modelling, Data Discovery, Open-Data Publication, Meta-data Management, Master & Persistence Data Management, Analytics, Reporting & Dashboarding, Geo-Visualization, Semi/unstructured Data Management, Integral Search and Navigation, Data Recording.

Our domain of interest in this report is focused on layers 3 to 4 of the EIP-SCC reference architecture.

## F.25 ESPRESSO

ESPRESSO (Exner, 2016) defines the sensing layer as an integrated layer where physical devices and connection drivers are included, while EIP-SCC distinguishes services in multiple layers. ESPRESSO considers only one layer that contains all IT enabled services

[56] European Innovation Partnership – Smart Cities & Communities (EIP-SCC), "Requirements specification for Urban Platforms", EIP-SCC Integrated Infrastructures action cluster – Urban Platform, technical report, v2.2 (consultation stage), 2016, https://smart-cities-marketplace.ec.europa.eu/sites/default/files/EIP_RequirementsSpecificationGLA%20V2-5.pdf

where all applications are merged together, although the visualization/analytics are split into the business layer. In all related reference architectures, Interoperability based on *open standards* also needs to be addressed with the objective of providing *open APIs*, *open Data* and open SDKs towards all directions. Services are focused on the end-users.

ESPRESSO is one of the underlying architecture approaches used in the definition of D1.2.

## F.26 SCC1 projects

There are a number of relevant earlier H2020 SCC1 Lighthouse projects with ICT architecture approaches, which are discussed in more detail in the appendix of D1.2. Here we discuss mySMARTLife[57] as one example of these.

The mySMARTLife architecture framework is composed of 4 major layers to support the interoperability for smart cities relying on Open Standardized Urban Platforms (OSUP). These layers are:

1. Northbound Interoperability: to enable high-level services by means of data exchange mechanisms under Open API, SDK, and data concepts.
2. Southbound Interoperability: to ensure the integration of data related to multiple and heterogeneous data sources. This level acts as a driver for data ingestion and harmonization (see original Data Management processes).
3. Semantic Interoperability: where a common data model and language is used to specify data flows. SensorThings API has been selected as a solution (e.g., as a data model) to represent the observations for real-time data type.
4. Interoperability between OSUP and existing data platforms: this layer connects the Open Geospatial Consortium (OGC) and Machine-to-Machine systems.

The framework presented in the following figure shows the mentioned components and a layered approach. Of specific interest here is the Interoperability layer which contains items of OpenSDK, OpenAPI, OpenData and MyData.

[57] https://www.mysmartlife.eu/

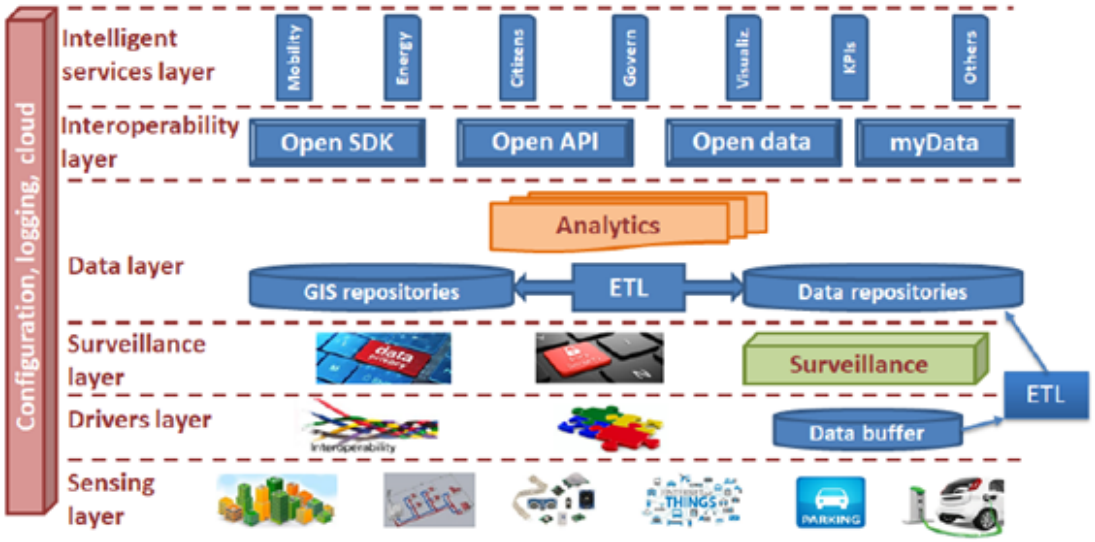


Figure: mySMARTLife Framework proposal (mySmartLife D1.16[58], page 39)

Other specifics of this framework:

1. Sensors can exchange data from city-to-city by using a specific protocol for communication
2. Services can exchange data from city-to-city by simply connecting APIs to the service (e.g., web service). Self-interpretation of existing data enables services to discover the available data relying on metadata presentational information
3. Linked data to provide data dictionaries (or catalogs) in the APIs to ensure third-party developers interoperability in terms of new added-value services
4. Openness: transparency, third-party developers can reuse data and open SDK to develop new APIs and therefore new services
5. Replicability: this framework is applied in different cities to mainly support interoperability
6. Scalability: achieved by data integration, data sources interoperability, new services and data storage.

[58] D1.16 in mySmartLife:
https://www.mysmartlife.eu/fileadmin/user_upload/Deliverables/D2.16_Open_Specifications_framework.pdf

# Appendix G – Related European Guidelines & Tools

## G.1 Using the European Interoperability Framework (EIF)

Following is an overview of the 47 *important* recommendations extracted from the European Interoperability Framework (EIF Contributors, 2020), which we have summarised for easy reference and communication in Appendix H. Implementation advice is also provided. Those partners who already use the simpler tools (level 1) should proceed to use these recommendations and also the other best practices and standards of DIIF Level 2.

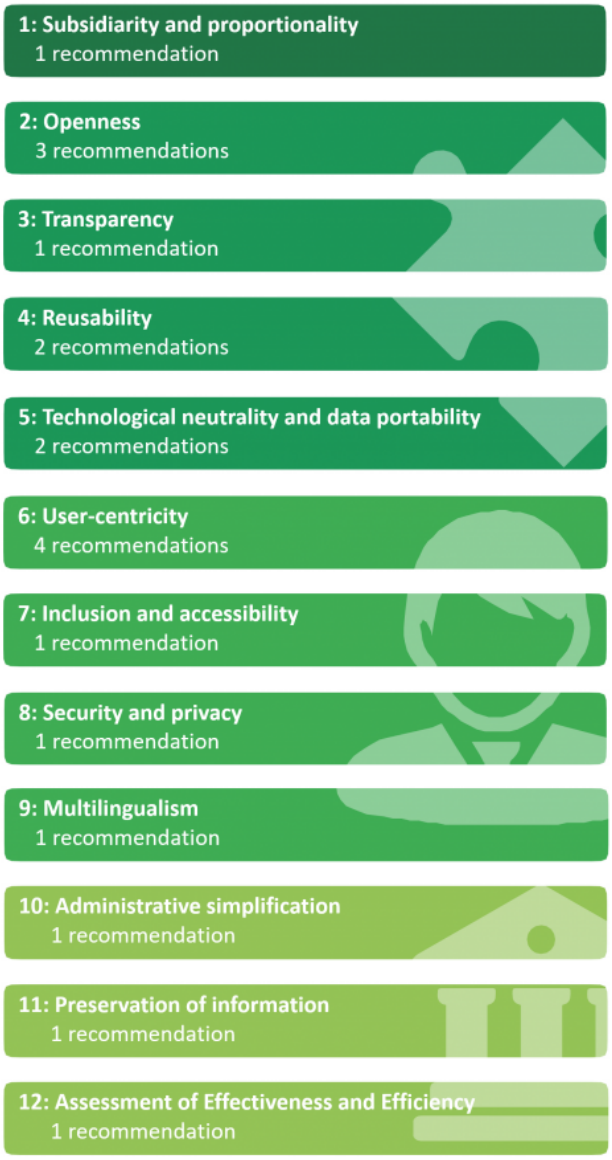


Figure: Overview of the EIF recommendation categories from Joinup (EIF Contributors, 2020)

The European Interoperability Framework for Smart Cities and Communities (EIF4SCC) is also being prepared[59] with the Joinup network at the time of writing. Consultations with experts are ongoing on EIF4SCC. It covers Technical and Semantic as well as other wider aspects such as Organisational Interoperability, and Cultural and Legal Interoperability; see figure below.


Figure: European Interoperability Framework for Smart Cities and Communities[60]

59 https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/news/eif4scc-smart-cities-communities

60 https://joinup.ec.europa.eu/collection/nifo-national-interoperability-framework-observatory/news/eif4scc-smart-cities-communities

EIF4SCC aims to support interoperability among the applications that address the multiple challenges within a smart city, such as housing, pandemics, energy and mobility. To enhance the well-being of citizens, interoperability plays an important role to bridge the fragmented services that may be offered within the different sectors as well as within and across communities and regions. This is taken into account in the EIF4SCC.

## G.2 Using the European Interoperability Reference Architecture (EIRA)

As we embrace a hybrid approach for building our framework (DIIF), we recommend EIF and EIRA in our framework's Level 2 as they have strong national and continental rigor (connecting local work upwards). EIF also has shortcomings which we prevail by innovations and by borrowing from other best practices as a hybrid approach.
Based on the EIF recommendations we mentioned in the previous section, a Reference Architecture (RA) has been proposed by the authors of EIF. The reference architecture, called EIRA (EIRA contributors, 2019), has different views:

1. Underlying principles
2. Organisational
3. Semantic
4. Legal
5. Technical
    a. Application
    b. Infrastructure

Realization of the achievement of organisational, legal, semantic and technical interoperability is the aim according to the "principles view" part of the archimate diagrams of the EIRA (EIRA contributors, 2019). Although the focus in +CityxChange has been more on the technical aspects, some vital organisational aspects have also been covered by best practices and standards in our hybrid framework (such as MSP and TCM), while EIRA is also part of our Level 2 (by referencing to it). EIRA principles:

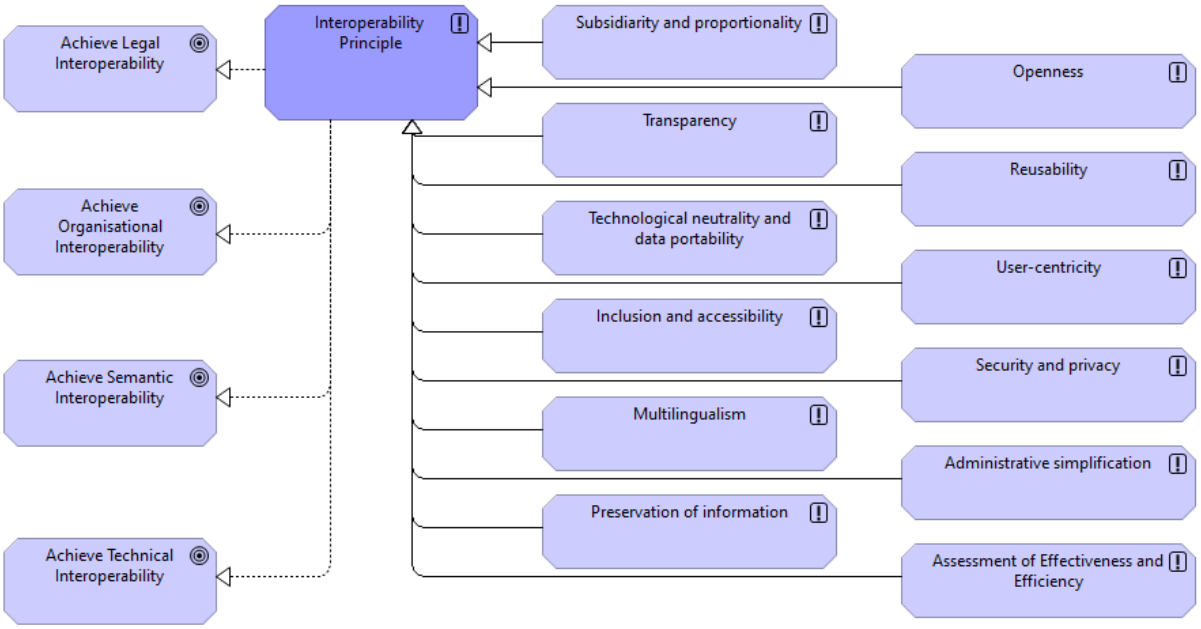


Figure - EIF Underlying Principles View, the basis of EIRA (source: (EIRA contributors, 2019))

The right part of the diagram summarises the gist of the rules mentioned in the previous section. More details regarding the recommendations and also the archimate file of EIRA can be found on the Joinup network[61] . The recommendations and diagrams produced by the consortium should be seen as resources completing/customising the general recommendations and diagrams, such as the ones mentioned, which are developed by support from bodies such as European Commission.

### G2.1 Assessment Tool for EIRA

Those partners who proceed to Level 2 interoperability and define a project to implement/use EIRA (EIRA contributors, 2019) (or do it as a part of other tasks/projects) and want to assess the interoperability achieved as a result, can refer to a specific tool designed for this.

EIF implementation advice provided in Appendix E can help partners in this regard before assessment, and then they can assess their EIF interoperability score using the tool. The tool comes in the form of an Excel sheet:

[61] https://joinup.ec.europa.eu/


Figure - EIRA (EIRA contributors, 2019) Assessment Tool

We have not yet come across a European smart city project doing and reporting full interoperability assessment according to EIF (and EIRA) using the aforementioned assessment tool.

## Appendix H – Implementation Advice for Partners on EIF

The European Interoperability Framework (EIF) (EIF Contributors, 2020) recommendations are not turned into implementable rules in EIF. Therefore we tried to address this for the consortium by introducing best practices in DIIF level 2 and by extracting and gathering rules from some of those as a sample. We have also provided advice for implementation,

one by one, in the following table.

Table: EIF Recommendations (Level 2) and our Implementation Advice (Level 3)

| **EIF Recommendation No.** | **Description by EIF (part of Level 2 of the hybrid DIIF, borrowed from EIF)** | **Our Advice for Implementation (part of DIIF Level 3)** |
|---|---|---|
| 1 | Ensure that national interoperability frameworks and interoperability strategies are aligned with the EIF and, if needed, tailor and extend them to address the national context and needs. | Communications with the Joinup network, ISA2 and national bodies such as those responsible for e-government can be the start point and specific national councils and groups may also exist which can be known through those entities. A common forum for constant communication and data meetings every three month with representatives from such bodies is recommended (can be common with other projects in the nation/continent). |
| 2 | Publish the data you own as open data unless certain restrictions apply. | The +CityxChange consortium is active on this, as part of its open data and open access strategy, participation in the open data pilot, and the Data Management Plan.. Automatic API Management, such as in FIWARE, should be explored for easier access and without time consuming permission seeking processes.<br>In specific cases and for data for which there is less motivation to be made open, the option of monetising APIs may be explored, albeit carefully given the open data ambition. |

| | | |
|---|---|---|
| 3 | Ensure a level playing field for open source software and demonstrate active and fair consideration of using open source software, taking into account the total cost of ownership of the solution. | Challenges to be addressed with Joinup network and there are limitations given private sector interests.<br>Open innovation made possible together with the community in WP3 can help with this and can distinguish a sub-community interested in open source development of solutions. |
| 4 | Give preference to open specifications, taking due account of the coverage of functional needs, maturity and market support and innovation. | Open specifications are introduced and encouraged in DIIF (including a number of continental resources. Please see sections 4 and 5).<br>Please refer to no. 2 in this table as well. |
| 5 | Ensure internal visibility and provide external interfaces for European public services. | Implementation of a knowledge management system (e.g. using media wiki, for the smart cities), and automatic API Management and discovery is encouraged. |
| 6 | Reuse and share solutions, and cooperate in the development of joint solutions when implementing European public services. | Collaboration with networks such as Joinup is encouraged (and contributing solutions/frameworks to the network). |
| 7 | Reuse and share information and data when implementing European public services, unless certain privacy or confidentiality restrictions apply. | Collaboration with networks such as Joinup is encouraged (and contributing solutions/framework to the network).<br>Please also refer to no. 2. |
| 8 | Do not impose any technological solutions on citizens, businesses and other administrations that are technology-specific or disproportionate to their real needs. | Agile method customised to each situation is encouraged (as for example taken in DIIF).<br>Federated exchange of information through APIs helps with this (rather than imposing integrated data models).<br>Efforts such as what is being done together with the community under +CityxChange WP3 also contributes to this. |

| | | |
|---|---|---|
| 9 | Ensure data portability, namely that data is easily transferable between systems and applications supporting the implementation and evolution of European public services without unjustified restrictions, if legally possible. | Federated exchange of information through APIs helps with this. |
| 10 | Use multiple channels to provide the European public service, to ensure that users can select the channel that best suits their needs. | Implementation budget is needed (and is usually justified when selling a service to an audience interested in specific channels), but not advised for all systems immediately (must be cost-effective, we cannot just do anything which is good due to - natural and inevitable - resource constraints).<br>Choice of good implementation tools can decrease the cost of multi-channel provision. |
| 11 | Provide a single point of contact in order to hide internal administrative complexity and facilitate users' access to European public services. | Automatic API Management and integration through FIWARE can help, plus Joinup/ISA2 collaborations (using which connecting efforts to the national and continental levels can be done) |
| 12 | Put in place mechanisms to involve users in analysis, design, assessment and further development of European public services. | Community Engagement and Innovation sub-programmes such as the one being done in +CityxChange WP3 can help |
| 13 | As far as possible under the legislation in force, ask users of European public services once-only and relevant-only information. | Should be orchestrated together with governments such that the open data made available by governments is used to not to ask repeatedly for existing data (and contributing data to the national open data needs to be done as well to support this goal).<br>Advice provided regarding no. 1 can help with this as well. |

| | | |
|---|---|---|
| 14 | Ensure that all European public services are accessible to all citizens, including persons with disabilities, the elderly and other disadvantaged groups. For digital public services, public administrations should comply with e-accessibility specifications that are widely recognised at European or international level. | Recommendation no. 10 and the advice provided regarding no. 10 can help with this.<br>Advice provided regarding no. 1 can help with this as well. |
| 15 | Define a common security and privacy framework and establish processes for public services to ensure secure and trustworthy data exchange between public administrations and in interactions with citizens and businesses. | Advice on no. 1 can help as it needs to be addressed in a wider scale together.<br>As a part of the solution, DLT data exchange work such as the one done by IOTA-+CityxChange and our assessment framework NovelSAM (part of DIIF Level 3) can help with this.<br>The 100+ API Implementation rules recommended in this report according to a best practice as a part of level 2 can help with this as well (see Appendix G). |
| 16 | Use information systems and technical architectures that cater for multilingualism when establishing a European public service. Decide on the level of multilingualism support based on the needs of the expected users. | Multiculturalism and localisation might be a better word, and from Unicode to localised APIs and services for time, date, language and so on should be used (can be located in major software development platforms and forums, and also where more specialisation is needed (e.g. GeoInformatics/ cartography Localisation), through networks such as Joinup and other introduced resources in sections 2 and 4). INSPIRE (Tóth & Smits, 2007) may help regarding spatial information (please refer to Appendix I, which contains more explanation and a link to a |

| | | |
|---|---|---|
| | | comprehensive document on using INSPIRE). |
| 17 | Simplify processes and use digital channels whenever appropriate for the delivery of European public services, to respond promptly and with high quality to users' requests and reduce the administrative burden on public administrations, businesses and citizens. | Digital Transformation and relevant process re-engineering and other transformational optimisations and changes for the organisation can be done using hybrid meta-frameworks such as TCM (Shams, & Kermanshah, 2018) (themselves pointing to numerous well-known methods, frameworks and best practices in a holistic way). |
| 18 | Formulate a long-term preservation policy for information related to European public services and especially for information that is exchanged across borders. | Advice on no. 1 can help such that our share is done.<br>This would often be part of city or national strategies. |
| 19 | Evaluate the effectiveness and efficiency of different interoperability solutions and technological options considering user needs, proportionality and balance between costs and benefits. | Can be done in a customised way in Level 3 of DIIF for every situation, such as the NovelSAM evaluation sub-framework of DIIF (for DLT services).<br>Advice for no. 1 should be used to bring together the different elements of what fulfils no. 19 in the continent (NovelSAM being part of it).<br>Criteria provided to compare interoperability frameworks in section 4 of the report can also help with no. 19. |
| 20 | Ensure holistic governance of interoperability activities across administrative levels and sectors. | Advice for no. 1 should be used. |
| 21 | Put in place processes to select relevant standards and specifications, evaluate them, monitor their implementation, check compliance and test their interoperability. | RI&D process in section 4 helps with this (as well as the shared experience of +CityxChange regarding the use of the processes. |

| | | |
|---|---|---|
| 22 | Use a structured, transparent, objective and common approach to assessing and selecting standards and specifications. Take into account relevant EU recommendations and seek to make the approach consistent across borders. | Advice for no. 1 should be used. DIIF itself helps with this, through which we have made progress towards helping with no. 22, which can be used by others as well. |
| 23 | Consult relevant catalogues of standards, specifications and guidelines at national and EU level, in accordance with your NIF and relevant DIFs, when procuring and developing ICT solutions. | Advice for no. 1 should be used. |
| 24 | Actively participate in standardisation work relevant to your needs to ensure your requirements are met. | Advice for no. 1 should be used. DIIF itself helps with this. |
| 25 | Ensure interoperability and coordination over time when operating and delivering integrated public services by putting in place the necessary governance structure. | DIIF itself helps with this already (and can be used beyond +CityxChange). Possible future development of DIIF Level 3 (and other aspects) can help. Advice for no. 1 should be used. |
| 26 | Establish interoperability agreements in all layers, complemented by operational agreements and change management procedures. | DIIF itself helps with this already (and can be used beyond +CityxChange). |
| 27 | Ensure that legislation is screened by means of 'interoperability checks', to identify any barriers to interoperability. When drafting legislation to establish a European public service, seek to make it consistent with relevant legislation, perform a 'digital check' and consider data protection requirements. | Advice for no. 1 should be used. |
| 28 | Document your business processes using commonly accepted modelling techniques and agree on how these | TCM (Shams, & Kermanshah, 2018) can help significantly. Advice for no. 1 should also be used. |

| | | |
|---|---|---|
| | processes should be aligned to deliver a European public service. | |
| 29 | Clarify and formalise your organisational relationships for establishing and operating European public services. | TCM (Shams, & Kermanshah, 2018) can help significantly.<br>Advice for no. 1 should also be used. |
| 30 | Perceive data and information as a public asset that should be appropriately generated, collected, managed, shared, protected and preserved. | Social Marketing methods can help with improving those perceptions.<br>Advice for no. 1 should also be used. |
| 31 | Put in place an information management strategy at the highest possible level to avoid fragmentation and duplication. Management of metadata, master data and reference data should be prioritised. | +CityxChange has a task on this, the report for which is reflected in D11.16 Data Management Plan (Ahlers et al., 2020). |
| 32 | Support the establishment of sector-specific and cross-sectoral communities that aim to create open information specifications and encourage relevant communities to share their results on national and European platforms. | Advice for no. 1 should be used. |
| 33 | Use open specifications, where available, to ensure technical interoperability when establishing European public services. | DIIF supports it, e.g. in its Level 2. |
| 34 | Use the conceptual model for European public services to design new services or re-engineer existing ones and reuse, whenever possible, existing service and data components. | Level 2 of DIIF, and in particular using EIRA of EIF, supports this. |
| 35 | Decide on a common scheme for interconnecting loosely coupled service components and put in place and maintain the necessary infrastructure for establishing and maintaining European public services. | API Catalog (Level 1) and relevant best practices in DIIF (its Level 2) help with this.<br>Partners need to progress in Level 1 and Level 2. The processes in section 4 helps with such progress. |

| | | |
|---|---|---|
| 36 | Develop a shared infrastructure of reusable services and information sources that can be used by all public administrations. | Advice for no. 1 should be used. |
| 37 | Make authoritative sources of information available to others while implementing access and control mechanisms to ensure security and privacy in accordance with the relevant legislation. | API Implementation rules extracted from a best practice and advised as a part of Level 2 of DIIF can help (Appendix I).<br>API specification and implementation standards (Level 2), and the APIs in +CityxChange API Catalog can help with achieving this. Implementation of API Catalog enhancements (Level 3) can help.<br>Advice for no. 1 should be used.<br>Collaboration on Joinup helps for example. |
| 38 | Develop interfaces with base registries and authoritative sources of information, publish the semantic and technical means and documentation needed for others to connect and reuse available information. | Please refer to advice for no. 37. |
| 39 | Match each base registry with appropriate metadata including the description of its content, service assurance and responsibilities, the type of master data it keeps, conditions of access and the relevant licences, terminology, a glossary, and information about any master data it uses from other base registries. | Using FIWARE Catalog and FIWARE data model can help (Level 2 of DIIF). Customisations regarding this can be done as part of Level 3 using the processes in section 4.<br>Please also refer to the advice (referenced to) for no. 31. |
| 40 | Create and follow data quality assurance plans for base registries and related master data. | Please refer to the advice (referenced to) for no. 31. |
| 41 | Establish procedures and processes to integrate the opening of data in your common business processes, working routines, and in the development of new information systems. | Government and EU incentive and support through Joinup can help. Then also customisations in line with Level 3 of DIIF can be done firm-by-firm or city-by-city. |

| | | |
|---|---|---|
| 42 | Publish open data in machine-readable, non-proprietary formats. Ensure that open data is accompanied by high quality, machine-readable metadata in non-proprietary formats, including a description of their content, the way data is collected and its level of quality and the licence terms under which it is made available. The use of common vocabularies for expressing metadata is recommended. | API Specification best practices (Level 2 of DIIF), the API Catalog (Level 1) and its enhancements (Level 3) help with this.<br>Level 2 and Level 3 implementations should be done by partners and existing API Catalog entries improved according to the training in data meetings and according to D1.3. |
| 43 | Communicate clearly the right to access and reuse open data. The legal regimes for facilitating access and reuse, such as licences, should be standardised as much as possible. | Preparing Data Contracts by partners can help. |
| 44 | Put in place catalogues of public services, public data, and interoperability solutions and use common models for describing them. | DIIF implementation, specially regarding APIs, helps with this (including the API Catalog). Please refer to section 4 for the processes. |
| 45 | Where useful and feasible to do so, use external information sources and services while developing European public services. | Joinup network and other national/continental resources should be used (please refer to Appendix F). |
| 46 | Consider the specific security and privacy requirements and identify measures for the provision of each public service according to risk management plans. | The 100+ API Implementation rules recommended in this report according to a best practice as a part of level 2 can help with this (see Appendix G). |
| 47 | Use trust services according to the Regulation on eID and Trust Services as mechanisms that ensure secure and protected data exchange in public services. | eIDAS for SMEs toolkit can be used.<br><br>For other resources please see their main web page. |

The tools and theoretical work surrounding EIF are being developed and the work here can be developed/used to be part of that.

# Appendix I – API Implementation Guides

Out of the API Technical and Data standard (v2, 2019, British Government), we mention the 33 API implementation rule categories and also mention the 100+ sub-rules extracted from these 33, for quick and easy reference of partners who are implementing APIs and want to ensure interoperability through the proper implementation of the APIs (although we do not aim to mention all the possible sub-rules, we have extracted important ones from the standard). The important rules extracted, which we advise partners to use for API implementation[62]:

1. Follow the Technology Code of Practice
    a. Open Access (where reasonable)
    b. Open process
    c. Avoiding duplicate work
    d. Consensus-based open process
    e. Government security policies and guidelines
2. Use RESTful
3. Use HTTPS
    a. Secure APIs using Transport Layer Security (TLS)
    b. Timely certificate renewal
4. Consider linking data (hypermedia)
    a. Use URIs (e.g. in API output)
5. Use JSON
    a. APIs to respond as a JSON object, not an array
    b. Document your JSON object well
    c. Use consistent grammar
    d. Avoid unpredictable object keys
6. To represent time and date
    a. Using the ISO 8601 standard
7. To represent a physical location
    a. Use the ETRS89 standard for Europe
    b. Use the World Geodetic System 1984 (WGS 84) for the rest of the world
    c. Use GeoJSON to exchange location info
8. Use Unicode for encoding
    a. Use Unicode Transformation Format (UTF-8)
9. How to respond to data requests

[62] Although we have gathered/extracted the rules and have made the rules concise, it should be mentioned that the copyright rules in the site also allow reuse with provision of reference to the source: API Technical and Data standard (v2, 2019, British Government). Used here under the applicable Open Government License, accessed 19/08/2020.

   a. Answer "requests", by sending only the information the user requires, not more data
10. Design data fields with user needs in mind
   a. consider needs arising from localisation and cultural differences, e.g. some cultures don't have first and last names
11. Let users download whole datasets in bulk
   a. Allow users download whole data unless there's restricted information
   b. Provide data update notification facility if you allow whole data download
12. Encourage users to keep local dataset copies up to date
   a. Let users download incremental changes (not whole data) when an update is available
13. When publishing bulk data
   a. Make data available using CSV (and also JSON), for bulk data, this allows user to be able to use a wider range of tools easier (without need for data conversion)
   b. Publish data to the national data repository (in UK: data.gov.uk)
14. Keep a log of requests for personal data
   a. If your API serves personal or sensitive data, log when the data is provided and to whom.
15. When to use open access
   a. Use open access API if you want to provide (almost) restriction-free API access and you do not need to identify your users
   b. You can still throttle your API
   c. Consider publishing to the national repository instead (in UK: data.gov.uk)
16. When to authenticate your API
   a. For fulfilling security requirements and authorisation
   b. For rate limiting/throttling
   c. For auditing
   d. For billing
   e. Note: to identify users only for rate limiting, you may not need to refresh user tokens very often as a stollen token is unlikely to threaten your service
   f. Note: for sensitive data (medical records, ...), API may require more than just authenticating an organisation token
17. To provide application-level authorisation
   a. Application-level authorisation is advised for APIs providing sensitive data unless you trust your consumers, e.g. a government department.
   b. Use OAuth 2.0, the open authorisation framework (specifically with the Client Credentials grant type), which gives each registered application an OAuth2 Bearer Token, to make API requests on the application's own behalf.
18. To provide user-level authorisation

a. Suitable for dealing with personal/sensitive data
b. Use OAuth 2.0 Scopes for more granular access control
c. OpenID Connect (OIDC), which builds on top of OAuth2, with its use of JSON Web Token (JWT), may be suitable in some cases such as a federated system.

19. For privacy and allow lists
    a. Use an allow list for the API to be private
    b. Do not add the IP addresses of the APIs you consume to allow list as APIs may be provided using Content Delivery Networks (CDNs) and scalable load balancers, which rely on dynamic allocation of IP and sharing.
    c. For the APIs you consume use an HTTPS egress proxy (instead of allow list).
20. Follow good practice for tokens and permissions
    a. Suitable refresh frequency and expiry period for user access tokens (to avoid vulnerabilities)
    b. Allow users to revoke authority
    c. If you suspect a token has been compromised, invalidate it and force a re-issue.
    d. Use Time-based One-Time Passwords (TOTP) to further secure APIs with application-level authorisation
    e. Use Multi-Factor Authentication (MFA) and Identity Verification (IV) to further secure APIs with user-level authorisation
    f. The tokens you provide should have the least permissions reasonable (in case compromised, there will be less trouble)
21. Monitor APIs for unusual activity
    a. Consider best practices of how to implement a monitoring strategy
    b. Consider best practice specifics of how to monitor the security status of networks and systems.
22. When naming and hosting your API
    a. Follow guidance on choosing a domain name.
    b. Names of APIs, namespaces and resources should:
        i. use nouns rather than verbs
        ii. be short, simple and clearly understandable
        iii. avoid technical or specialist terms where possible
        iv. use hyphens rather than underscores as separators
        For example: [api-name].api.gov.ie
23. Avoid the use of namespaces
    a. Each API should have its own domain, which helps with avoiding API sprawl and simplifies versioning.

24. When you need to provide multiple APIs from the same domain

   a. You do this when you deploy common services across them: common management, common authentication and common security approaches
   b. Differentiate them through namespaces which reflect the function of government being offered by the API

25. When using sub-resources
   a. Should be no more than three levels deep: /resource/id/sub-resource/id/sub-sub-resource.
26. When using query arguments
   a. Use path parameters to identify specific resource(s), e.g.: example, /users/1.
   b. Only allow query strings to be used in GET requests for filtering the values returned from an individual resource, e.g.: /users?state=active or /users?page=2.
   c. Never use query strings in GET requests for identification purposes, e.g. avoid: /users?id=1.
   d. Query strings should not be used for defining the behaviour of API: /users?action=getUser&id=1.
27. When iterating your API
   a. Make backwards compatible changes where reasonably possible
   b. Make new endpoint available for significant changes
   c. Provide notice for deprecated endpoint
28. When making a backwards incompatible change
   a. Increment a version number in the URL or the HTTP header (start with /v1/ and increment with whole numbers)
   b. Support both old and new endpoints in parallel for a suitable time period
   c. Tell users of API how to update their data validation, e.g. tell them when a field is not going to be present
   d. To simplify a complex object structure by merging, make the new object available at a new endpoint, e.g.:
      Merge *users data* and *accounts data* from:
      /v1/users/123 and /v1/accounts/123
      To produce:
      /v1/consolidated-account/123
29. Set clear deprecation policies
   a. Announce deprecation in HTTP responses e.g. by using a 'Warning' header
   b. Consider direct contact, e.g. using email
   c. Express how much time users have to upgrade
30. Provide users with a test service (Sandbox)
   a. Read only APIs do not necessarily need to offer a test service
   b. Mimic the real service in test service and be vary of user needs (consult) and the implications of exact simulations (load, risk, ...)

31. Test your API's compliance
    a. Provide sample test data
32. Test your API's performance and scalability
    a. For highly cacheable open data access APIs, a well-configured Content Delivery Network (CDN) may provide sufficient scalability.
    b. Test the capacity of your APIs
    c. Make sure users can test your full API up to the quotas you have set.
    d. Where the API delivers personal or private information you, as the data controller, must provide sufficient timeouts on any cached information in your delivery network.
33. Document your API
    a. Use the OpenAPI 3 Specification
    b. Consider guidance on how to document APIs and how to write API reference documentation
    c. Provide sample code to illustrate how to consume the API and the responses the consumer can expect from the API
    d. According to this standard, the API documentation should include:
        i. Overview: what the API does, who it might be used by, ...
        ii. Business and data rules - under what circumstances is data available / not available
        iii. Error scenarios
        iv. Details on the test service - how to use it and how to simulate the various success and error scenarios
        v. Request and response parameters, including information on meaning, data type and any other constraints.
        vi. Rules on information handling, incident management and risk management
        vii. Method of authentication (including how it impacts service interoperability, single sign-on, and rate-limiting)
        viii. Authorisation rules
        ix. Versioning information and design changes
        x. Availability, latency, ownership, deprecation policies and status capability
        xi. Approach to backwards compatibility
        xii. Information on configuring the API
        xiii. Security information
        xiv. Cost of use

It is understood that some of the partners have internal resources allocated to a number of the above besides other tasks. In general, implementation can start from simpler systems with less number of best practices adhered to (where reasonable), and develop the systems towards implementing more and more of the best practices relevant, in an evolutionary way (using methods such as Agile or Prince2 Agile).

# Appendix J – Innovations Table

This section summarises the innovations provided by this Deliverable and links them to the relevant aspects of the GA.

| No. | Innovation Title | Importance | Consortium Agreement Relation Guide* | D1.3 Sections |
|---|---|---|---|---|
| 1 | +CxC Data Integration and Interoperability Framework (design as a whole) | Novel and more practical framework (e.g. compared to European Interoperability Framework), with Agile rationale, customised for +CxC | 1, 2, 3, 4, 5, 6, 7, 8, 9 | Section 3 |
| 2 | API Catalogue and its 69 APIs | Easy-to-use and simple-to-start-with tool and sub-framework agreed with partners to be good for capturing +CxC API info<br>69 +CxC ecosystem API entries added. The entries are used in D1.2 use cases in T1.1. | 2, 3, 5, 6 | Section 5 and Appendix G |
| 3 | Cross-partner Data-related Requirements Table | Easy-to-use and simple-to-start-with tool and sub-framework agreed with partners to be good in absence of PMO facilities and P3M etc. best practices for facilitating implementation of the framework in 1 | 3, 6, 7 | Section 4 |
| 4 | Level 2 of the Framework | Chosen best practices, standards and resources introduced directly in the Level-2 plus other ones mentioned in D1.3 (inner circle and wider circle of chosen resources), which can ensure data integration and interoperability in a more advanced level | 3, 4, 5, 6, 7, 8 | Sections 3, 4 and Appendices |
| 5 | Level 3 of the Framework | Integrated Program from all the mentioned resources including NovelSAM, API Catalogue Enhancements and Framework | 1, 2, 3, 4, 5, 6, 7, 8, 9 | Sections 3, 4, 5 and 6 and Appendices |

| | | | | |
|---|---|---|---|---|
| | | RI&D and Implementation Processes, and other sub-frameworks which can be produced if we have resources (using Level 2 resources plus innovations to create: advanced general purpose tools/sub-frameworks + advanced customisation for +CxC) | | |
| 6 | NovelSAM | Novel DLT Service Assessment Model for +CxC and others (base service assessment model to be published, NovelSAM paper itself being prepared and to be published) | 8, 9 | Section 6 and Appendices |
| 7 | +CxC Data Integration and Interoperability Framework RI&D and Implementation Processes | Processes for use of other follower cities and smart city projects to help with reproduction of the results | 3, 6, 7 | Section 4 |

* Guide structure (reflecting grant agreement):

1. On the methodology and D1.3 rationale
2. On the coordination with the architecture developed in T1.1
3. How to ensure data integration and interoperability of ICT systems and services, including software platforms and tools, data repositories, and IoT devices?
4. Identifying open standards for data vocabularies and data models in the smart city domain
5. Integrating corresponding API specifications
6. Ensuring agreement on open standards between service providers involved in demo projects for interoperability
7. Supporting data flow between partners and towards the KPI collection into the monitoring platform in WP7
8. Examining securing distributed data integration between various services in the ICT ecosystem: the potential of using IOTA and other DLTs for data transfer
9. Fall-back security and exchange mechanisms for DLT (ensuring data integrity and confidentiality)