



H2020 - INDUSTRIAL LEADERSHIP - Information and Communication Technologies (ICT)
ICT-14-2016-2017: Big Data PPP: cross-sectorial and cross-lingual data integration and experimentation



ICARUS:

“Aviation-driven Data Value Chain for Diversified Global and Local Operations”

D3.3 – Architecture, Core Data and Value Added Services Bundles Specifications- v2.00

| | | | |
|---------------------|---------------------------------|------------------------|--------|
| Workpackage: | WP3 – ICARUS Platform Design | | |
| Authors: | UBITECH, SUITE5, SILO, ENG, UCY | | |
| Status: | Final | Classification: | Public |
| Date: | 17/07/2019 | Version: | 1.00 |

Disclaimer:









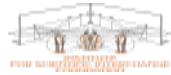



The ICARUS project is co-funded by the Horizon 2020 Programme of the European Union. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

© Copyright in this document remains vested with the ICARUS Partners.

ICARUS Project Profile

| | |
|-----------------------------|--|
| Grant Agreement No.: | 780792 |
| Acronym: | ICARUS |
| Title: | Aviation-driven Data Value Chain for Diversified Global and Local Operations |
| URL: | http://www.icarus2020.aero |
| Start Date: | 01/01/2018 |
| Duration: | 36 months |

Partners

| | | |
|---|---|----------------|
|  | UBITECH (UBITECH) | Greece |
|  | ENGINEERING - INGEGNERIA INFORMATICA SPA (ENG) | Italy |
|  | PACE Aerospace Engineering and Information Technology GmbH (PACE) | Germany |
|  | SUITE5 DATA INTELLIGENCE SOLUTIONS LIMITED (SUITE5) | Cyprus |
|  | UNIVERSITY OF CYPRUS (UCY) | Cyprus |
|  | CINECA CONSORZIO INTERUNIVERSITARIO (CINECA) | Italy |
|  | OAG Aviation Worldwide LTD (OAG) | United Kingdom |
|  | SingularLOGIC S.A. (SILO) | Greece |
|  | ISTITUTO PER L'INTERSCAMBIO SCIENTIFICO (ISI) | Italy |
|  | CELLOCK LTD (CELLOCK) | Cyprus |
|  | ATHENS INTERNATIONAL AIRPORT S.A (AIA) | Greece |
|  | TXT e-solutions SpA (TXT) – 3 rd party of PACE | Italy |

Document History

| Version | Date | Author (Partner) | Remarks |
|-------------|------------|--|--|
| 0.10 | 03/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH) | Initial Table of Contents |
| 0.20 | 06/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH) | Initial outline of Sections 2 and 3 |
| 0.30 | 10/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5) | Initial contribution to section 2 |
| 0.40 | 12/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5) | Updated contribution to section 2, Initial contribution to section 3 |
| 0.50 | 14/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5) | Updated contribution to section 3: 3.3, 3.6, 3.7, 3.8, 3.11, 3.13, 3.14, 3.19 by UBITECH, 3.4, 3.5, 3.10, 3.15 by Suite5 |
| 0.60 | 17/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5), Susanna Bonura, Domenico Messina (ENG), Dimosthenis Stefanidis, Loukas Pouis (UCY), Pavlos Lampadaris, Tasos Violetis (SILO) | Updated contributions to sections: 3.3, 3.6, 3.7, 3.8, 3.11, 3.13, 3.14, 3.19 by UBITECH, 3.4, 3.5, 3.10, 3.15 by Suite5, 3.2, 3.9, 3.12 by SILO, 3.17, 3.18, 3.20 by ENG, 3.16, 3.21, 3.22 by UCY |
| 0.70 | 20/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5), Susanna Bonura, Domenico Messina (ENG), Dimosthenis Stefanidis, Loukas Pouis (UCY), Pavlos Lampadaris, Tasos Violetis (SILO) | Updated contributions to sections: 3.3, 3.6, 3.7, 3.8, 3.11, 3.13, 3.14, 3.19 by UBITECH, 3.4, 3.5, 3.10, 3.15 by Suite5, 3.2, 3.9, 3.12 by SILO, 3.17, 3.18, 3.20 by ENG, 3.16, 3.21, 3.22 by UCY |
| 0.80 | 24/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Fenareti Lampathaki, Evmorfia Biliri (Suite5), Susanna Bonura, Domenico Messina (ENG), Dimosthenis Stefanidis, Loukas Pouis (UCY), Pavlos Lampadaris, Tasos Violetis (SILO) | Updated contributions to sections: 3.3, 3.6, 3.7, 3.8, 3.11, 3.13, 3.14, 3.19 by UBITECH, 3.4, 3.5, 3.10, 3.15 by Suite5, 3.2, 3.9, 3.12 by SILO, 3.17, 3.18, 3.20 by ENG, 3.16, 3.21, 3.22 by UCY |
| 0.90 | 28/06/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH) | Updated full draft circulated for internal review |
| 0.90_TXT | 08/07/2019 | Michele Sesana (TXT) | Internal review |
| 0.90_CINECA | 09/07/2019 | Giorgio Pedrazzi (CINECA) | Internal review |
| 0.95 | 16/07/2019 | Dimitrios Miltiadou, Konstantinos Perakis (UBITECH) | Updated version addressing comments received during the internal review process |
| 1.0 | 17/07/2019 | Fenareti Lampathaki (Suite5), Dimitris Alexandrou (UBITECH) | Final version for submission to the EC |

Executive Summary

The document at hand, entitled “Architecture, Core Data and Value Added Services Bundles Specifications-v2.00” constitutes a report of the efforts and the produced results of Tasks T3.1 “Technology Requirements”, T3.2 “Demonstrator User Requirements”, T3.3 “Platform Architecture Design and APIs Specifications”, T3.4 “Core Data Service Bundles In-depth Design” and T3.5 “ICARUS Added Value Services Design” of WP3. The purpose of this deliverable is to deliver the complementary documentation of the architecture of the ICARUS platform and the updated documentation of the components of the platform with regard to their functionalities and their offered interfaces. Within this context, the scope of the current report can be described in the following axes:

- To present a comprehensive documentation of the architecture of the integrated ICARUS platform. A brief description of each component is presented focusing on their positioning within the platform’s architecture. For each component, the platform functionalities that are undertaken by this component are described and the component’s interactions with the rest of the components for the realisation of these functionalities is documented.
- To provide the updated documentation of the components of the ICARUS platform. For each component of the integrated ICARUS platform, the core functionalities that the component offers are described. In addition to this, the involvement of each component in the ICARUS platform’s services and in the designed platform’s workflows is presented. Furthermore, for each component, the interactions with the rest of the components, as well as the interfaces that are offered in order to facilitate the required exchange of information, are presented. Finally, the technical details of these interfaces is documented.

The current deliverable presents the updated and supplementary documentation of the architecture of the integrated ICARUS platform and of all the components of the platform. It should be noted though that all tasks of WP3 remain active until M32 according to the ICARUS Description of Action. Thus, the design of the ICARUS platform’s architecture, as well as the design and specifications of the components of the architecture, will receive the necessary updates and refinements based on further identified functional requirements that will be translated into technical requirements, originating mainly from the evaluation and feedback received from the demonstrator partners. The upcoming versions of this deliverable, namely D3.4 and D3.5, will provide the necessary documentation of the aforementioned changes.

Table of Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 11 |
| 1.1 | Purpose | 11 |
| 1.2 | Document Approach..... | 12 |
| 1.3 | Relation to other ICARUS Results | 12 |
| 1.4 | Structure | 13 |
| 2 | ICARUS Platform Architecture | 15 |
| 3 | ICARUS Components Designs | 23 |
| 3.1 | Overview | 23 |
| 3.2 | Anonymiser | 23 |
| 3.2.1 | Services Outline | 23 |
| 3.2.2 | Interfaces | 24 |
| 3.3 | Cleanser..... | 26 |
| 3.3.1 | Services Outline | 26 |
| 3.3.2 | Interfaces | 27 |
| 3.4 | Mapper..... | 29 |
| 3.4.1 | Services Outline | 29 |
| 3.4.2 | Interfaces | 30 |
| 3.5 | Wallet Manager | 34 |
| 3.5.1 | Services Outline | 34 |
| 3.5.2 | Interfaces | 34 |
| 3.6 | Encryption Manager | 34 |
| 3.6.1 | Services Outline | 34 |
| 3.6.2 | Interfaces | 35 |
| 3.7 | Decryption Manager | 37 |
| 3.7.1 | Services Outline | 37 |
| 3.7.2 | Interfaces | 38 |
| 3.8 | Key-Pair Administrator | 39 |
| 3.8.1 | Services Outline | 39 |
| 3.8.2 | Interfaces | 40 |
| 3.9 | Data Handler | 41 |

| | | |
|-------------|--|------------|
| 3.9.1 | Services Outline | 41 |
| 3.9.2 | Interfaces | 42 |
| 3.10 | Data License and Agreement Manager | 51 |
| 3.10.1 | Services Outline | 51 |
| 3.10.2 | Interfaces | 52 |
| 3.11 | Policy Manager | 58 |
| 3.11.1 | Services Outline | 58 |
| 3.11.2 | Interfaces | 60 |
| 3.12 | ICARUS Storage and Indexing | 74 |
| 3.12.1 | Services Outline | 74 |
| 3.12.2 | Interfaces | 75 |
| 3.13 | Master Controller | 75 |
| 3.13.1 | Services Outline | 75 |
| 3.13.2 | Interfaces | 76 |
| 3.14 | OnPremise Worker and SecureSpace Worker | 78 |
| 3.14.1 | Services Outline | 78 |
| 3.14.2 | Interfaces | 79 |
| 3.15 | Query Explorer | 82 |
| 3.15.1 | Services Outline | 82 |
| 3.15.2 | Interfaces | 83 |
| 3.16 | Recommender | 86 |
| 3.16.1 | Services Outline | 86 |
| 3.16.2 | Interfaces | 86 |
| 3.17 | Analytics and Visualisation Workbench | 87 |
| 3.17.1 | Services Outline | 87 |
| 3.17.2 | Interfaces | 89 |
| 3.18 | BDA Application Catalogue | 98 |
| 3.18.1 | Services Outline | 98 |
| 3.18.2 | Interfaces | 99 |
| 3.19 | Resource Orchestrator | 106 |
| 3.19.1 | Services Outline | 106 |

| | |
|---|------------|
| 3.19.2 Interfaces..... | 107 |
| 3.20 Jobs Scheduler and Execution Engine | 109 |
| 3.20.1 Services Outline | 109 |
| 3.20.2 Interfaces..... | 109 |
| 3.21 Notification Manager..... | 113 |
| 3.21.1 Services Outline | 113 |
| 3.21.2 Interfaces..... | 114 |
| 3.22 Usage Analytics..... | 119 |
| 3.22.1 Services Outline | 119 |
| 3.22.2 Interfaces..... | 120 |
| 4 Conclusions & Next Steps | 134 |

List of Figures

| | |
|--|----|
| Figure 1-1: Relation to other ICARUS Work Packages | 13 |
| Figure 2-1: ICARUS high-level architecture..... | 16 |

List of Tables

| | |
|--|----|
| Table 3-1: Anonymiser - initiate process | 24 |
| Table 3-2: Anonymiser - process status..... | 25 |
| Table 3-3: Cleanser - initiate process | 27 |
| Table 3-4: Cleanser - process status | 28 |
| Table 3-5: Cleanser - obtain log records..... | 28 |
| Table 3-6: Mapper - calculate mapping..... | 30 |
| Table 3-7: Mapper - save mapping..... | 31 |
| Table 3-8: Mapper - save mapping and train model | 32 |
| Table 3-9: Mapper - initiate mapping process..... | 32 |
| Table 3-10: Mapper - process status | 33 |
| Table 3-11: Encryption Manager -initiate process | 35 |
| Table 3-12: Encryption Manager - process status | 36 |
| Table 3-13: Encryption Manager - receive decryption request..... | 36 |
| Table 3-14: Decryption Manager - initiate process | 38 |
| Table 3-15: Decryption Manager - check connection response | 38 |
| Table 3-16: Decryption Manager -process decryption response | 39 |
| Table 3-17: Key Pair Administrator - connection request | 40 |
| Table 3-18: Key Pair Administrator - check data access rights | 41 |
| Table 3-19: Data Handler - create new data preparation job | 42 |
| Table 3-20: Data Handler - get data preparation job | 43 |
| Table 3-21: Data Handler - modify data preparation job | 44 |
| Table 3-22: Data Handler - create data preparation instructions | 45 |
| Table 3-23: Data Handler - get data preparation instructions | 46 |
| Table 3-24: Data Handler - execute data preparation instructions..... | 47 |
| Table 3-25: Data Handler - get data preparation status..... | 47 |
| Table 3-26: Data Handler - upload data sample..... | 48 |
| Table 3-27: Data Handler - upload data | 48 |
| Table 3-28: Data Handler - download data | 49 |
| Table 3-29: Data Handler - transfer data..... | 49 |
| Table 3-30: Data Handler - add metadata | 50 |
| Table 3-31: Data Handler - get metadata | 50 |
| Table 3-32: Data Handler - update metadata..... | 51 |
| Table 3-33: Data License and Agreement Manager - request to buy data asset | 52 |
| Table 3-34: Data License and Agreement Manager - retrieve request to buy data asset..... | 53 |
| Table 3-35: Data License and Agreement Manager - reject request to buy data asset | 54 |
| Table 3-36: Data License and Agreement Manager - accept request to buy data asset..... | 54 |
| Table 3-37: Data License and Agreement Manager - mark buy data asset as paid..... | 55 |
| Table 3-38: Data License and Agreement Manager - create new contract..... | 55 |
| Table 3-39: Data License and Agreement Manager - get contract..... | 56 |
| Table 3-40: Data License and Agreement Manager - update contract..... | 57 |
| Table 3-41: Policy Manager - create organisation..... | 60 |

| | |
|---|-----|
| Table 3-42: Policy Manager - get organisation | 61 |
| Table 3-43: Policy Manager - get all organisations..... | 62 |
| Table 3-44: Policy Manager - update organisation..... | 63 |
| Table 3-45: Policy Manager - suspend organisation..... | 64 |
| Table 3-46: Policy Manager - invite users to organisation | 65 |
| Table 3-47: Policy Manager - get users of organisation | 65 |
| Table 3-48: Policy Manager - create user | 66 |
| Table 3-49: Policy Manager - update user..... | 67 |
| Table 3-50: Policy Manager - get user | 68 |
| Table 3-51: Policy Manager - suspend user | 69 |
| Table 3-52: Policy Manager - login | 70 |
| Table 3-53: Policy Manager - logout..... | 70 |
| Table 3-54: Policy Manager - create access policy | 71 |
| Table 3-55: Policy Manager - modify access policy | 72 |
| Table 3-56: Policy Manager - delete access policy | 73 |
| Table 3-57: Policy Manager - authorise access request | 73 |
| Table 3-58: Policy Manager - access control filter..... | 74 |
| Table 3-59: Master Controller - receive instructions..... | 76 |
| Table 3-60: Master Controller - receive job status..... | 77 |
| Table 3-61: Master Controller - transfer data | 77 |
| Table 3-62: Master Controller - upload results | 78 |
| Table 3-63: On Premise Worker and SecureSpace Worker - receive instructions | 79 |
| Table 3-64: On Premise Worker and SecureSpace Worker - retrieve job status..... | 80 |
| Table 3-65: On Premise Worker and SecureSpace Worker - receive task status | 81 |
| Table 3-66: On Premise Worker and SecureSpace Worker - upload results | 81 |
| Table 3-67: On Premise Worker and SecureSpace Worker – receive data | 82 |
| Table 3-68: Query Explorer - create and execute query..... | 83 |
| Table 3-69: Query Explorer - get query definition..... | 84 |
| Table 3-70: Query Explorer - get updated query results | 84 |
| Table 3-71: Query Explorer - delete query | 85 |
| Table 3-72: Query Explorer - get query history | 85 |
| Table 3-73: Recommender - data assets recommendations..... | 87 |
| Table 3-74: Analytics and Visualisation Workbench – register algorithm | 89 |
| Table 3-75: Analytics and Visualisation Workbench – get, delete algorithm | 90 |
| Table 3-76: Analytics and Visualisation Workbench – register application..... | 91 |
| Table 3-77: Analytics and Visualisation Workbench – get applications | 92 |
| Table 3-78: Analytics and Visualisation Workbench – get application..... | 94 |
| Table 3-79: Analytics and Visualisation Workbench – delete application..... | 96 |
| Table 3-80: Analytics and Visualisation Workbench – schedule job..... | 96 |
| Table 3-81: Analytics and Visualisation Workbench – add job..... | 97 |
| Table 3-82: Analytics and Visualisation Workbench – get all jobs | 98 |
| Table 3-83: BDA Application Catalogue - application creation..... | 99 |
| Table 3-84: BDA Application Catalogue - get application | 100 |
| Table 3-85: BDA Application Catalogue - update application..... | 102 |
| Table 3-86: BDA Application Catalogue - delete application..... | 104 |
| Table 3-87: BDA Application Catalogue - get all applications..... | 105 |
| Table 3-88: Resource Orchestrator - deploy secure and private space..... | 107 |

| | |
|---|-----|
| Table 3-89: Resource Orchestrator - stop secure and private space..... | 107 |
| Table 3-90: Resource Orchestrator - deploy services on secure and private space | 108 |
| Table 3-91: Jobs Scheduler and Execution Engine – get entry | 110 |
| Table 3-92: Jobs Scheduler and Execution Engine - update entry..... | 110 |
| Table 3-93: Jobs Scheduler and Execution Engine - delete entry..... | 111 |
| Table 3-94: Jobs Scheduler and Execution Engine - add entry | 111 |
| Table 3-95: Jobs Scheduler and Execution Engine - get all entries..... | 112 |
| Table 3-96: Notification Manager - list retrieval | 114 |
| Table 3-97: Notification Manager - single notification retrieval | 115 |
| Table 3-98: Notification Manager - mark notification as seen | 116 |
| Table 3-99: Notification Manager - mark all notifications as seen..... | 116 |
| Table 3-100: Notification Manager - delete notification | 117 |
| Table 3-101: : Notification Manager - new data asset notification | 117 |
| Table 3-102: Notification Manager - data asset update notification | 117 |
| Table 3-103: Notification Manager - data asset request notification | 118 |
| Table 3-104: Notification Manager - data asset draft contract notification..... | 118 |
| Table 3-105: Notification Manager – job status update notification | 119 |
| Table 3-106: Usage Analytics - user registration | 121 |
| Table 3-107: Usage Analytics - user login | 121 |
| Table 3-108: Usage Analytics - user logout | 121 |
| Table 3-109: Usage Analytics - asset appeared in search..... | 122 |
| Table 3-110: Usage Analytics - asset viewed | 122 |
| Table 3-111: Usage Analytics - asset starred..... | 123 |
| Table 3-112: Usage Analytics - asset un-starred..... | 123 |
| Table 3-113: Usage Analytics - asset requested | 124 |
| Table 3-114: Usage Analytics - asset request rejected | 124 |
| Table 3-115: Usage Analytics - asset purchased..... | 124 |
| Table 3-116: Usage Analytics - asset created | 125 |
| Table 3-117: Usage Analytics - algorithm utilised | 125 |
| Table 3-118: Usage Analytics - visualisation utilised | 126 |
| Table 3-119: Usage Analytics - vm started | 126 |
| Table 3-120: Usage Analytics - vm stopped..... | 127 |
| Table 3-121: Usage Analytics - new analytics job..... | 127 |
| Table 3-122: Usage Analytics - analytics job success..... | 128 |
| Table 3-123: Usage Analytics - analytics job failed..... | 128 |
| Table 3-124: Usage Analytics - general assets statistics | 129 |
| Table 3-125: Usage Analytics - private assets statistics..... | 129 |
| Table 3-126: Usage Analytics - user private statistics..... | 130 |
| Table 3-127: Usage Analytics - admin private statistics | 132 |

1 Introduction

1.1 Purpose

The scope of the ICARUS deliverable D3.3 “Architecture, Core Data and Value Added Services Bundles Specifications-v2.00” is to document the efforts carried out within the context of all tasks of WP3, namely T3.1 “Technology Requirements”, T3.2 “Demonstrator User Requirements”, T3.3 “Platform Architecture Design and APIs Specifications”, T3.4 “Core Data Service Bundles In-depth Design” and T3.5 “ICARUS Added Value Services Design”.

The deliverable D3.3 is prepared in accordance with the ICARUS Description of Action and will provide the incremental updates of the documentation of the components of the ICARUS architecture. The deliverable D3.3 is building upon the outcomes of the deliverable D3.1 “ICARUS Architecture, APIs Specifications and Technical and User Requirements” in which the first version of the conceptual architecture of the integrated ICARUS platform was delivered, as well as the deliverable D3.2 “Core Data Service Bundles and Value Added Services Designs” in which the ICARUS platform’s workflows and the design of the services of the platform were presented, in order to provide the updated documentation of the functionalities and the interfaces of the ICARUS architecture components.

In this context, the scope of the current deliverable is:

- To provide the complementary documentation of the architecture of the integrated ICARUS platform supplementing the information documented in deliverable D3.1. Following the same approach as with deliverable D3.1, the ICARUS architecture is presented, describing how each component is involved in a specific functionality of the platform. It should be noted at this point that in this document there was no differentiation on the architecture as documented in D3.1, however the provided documentation is focusing on the positioning of the updated components within the architecture and the interactions between them are highlighted.
- To document the updated detailed descriptions of the components of the ICARUS platform, outlining the core functionalities of each component, their interactions with the rest of the components for the realisation of the ICARUS platform’s workflows and the interfaces that they offer towards this aim. Furthermore, the technical details of the interfaces that are offered by each component are documented.

The ICARUS deliverable D3.3 presents the updated documentation on the design and specifications of the ICARUS platform’s components towards the successful implementation of the designed workflows. However, all the tasks of WP3 remain active until M32 according to the ICARUS Description of Action and during this period, the identification and analysis of additional functional and non-functional requirements, as well as their translation into

technical requirements, is a living process and the design and specifications of the ICARUS platform's architecture and components will be constantly updated and documented in the upcoming versions of this deliverable.

1.2 Document Approach

The current deliverable follows a systematic and comprehensive approach in order to present the outcomes and the knowledge extracted from the work performed in all tasks of WP3.

At first, the complementary documentation of the ICARUS platform architecture is presented. In this documentation, the positioning of each component of the architecture is highlighted, the platform functionalities that each component is involved and the interactions between the components for the realisation of these functionalities, as it is also depicted in the designed ICARUS workflows, is presented.

Following the documentation of the ICARUS platform architecture, the updated documentation of the components of the platform is presented. For each component, the core functionalities that it is offering are highlighted. Furthermore, their involvement in the ICARUS platform services is presented and the interactions of each component with the rest of the components of the platform for each designed ICARUS platform workflow is documented. Finally, for each component, the technical details of the interfaces that are facilitating the interactions of the component with the other components are documented.

1.3 Relation to other ICARUS Results

The ICARUS Deliverable D3.3 is released in the scope of the WP3 "ICARUS Platform Design" activities and reports the efforts undertaken within the context of T3.1 "Technology Requirements", T3.2 "Demonstrator User Requirements", T3.3 "Platform Architecture Design and APIs Specifications", T3.4 "Core Data Service Bundles In-depth Design" and T3.5 "ICARUS Added Value Services Design" of WP3.

As depicted in **Error! Reference source not found.**, the initial outcomes of T3.1 and T3.2 provided the input to T3.3 in order to formulate the initial version of the integrated ICARUS platform and the outcomes of T3.3 were provided as input to both T3.4 and T3.5. The outcomes of the T3.4 and T3.5 also provided input to T3.3 and triggered the necessary updates in the design and APIs specifications of the components of the ICARUS platform.

As Tasks T3.1 and T3.2 will be constantly updated as the project evolves in order to follow the project's advancements, the updated outcomes will be also provided as input to T3.3, T3.4 and T3.5 in order to formulate the updated ICARUS platform architecture, as well as the

updated designs of the Core Data Service Bundles and the Added Value Services that will be documented in the upcoming versions of the WP3 deliverable series.

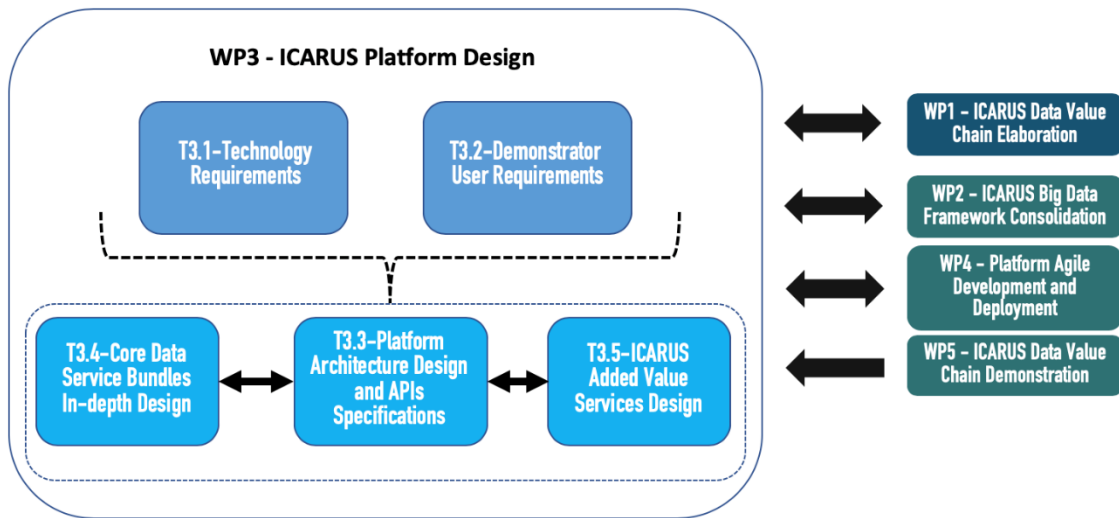


Figure 1-1: Relation to other ICARUS Work Packages

D3.3 builds on the outcomes of D3.1 and D3.2, updating them as necessary to reflect the latest project advancements across all Work Packages, taking into consideration the ICARUS platform development progress and feedback from WP4 “Platform Agile Development and Deployment”.

Furthermore, D3.3 and WP3 are directly related to the outcomes of: (a) WP1 “ICARUS Data Value Chain Elaboration” with regard to the ICARUS methodology, the ICARUS Minimum Viable Product (MVP), and (b) WP2 “ICARUS Big Data Framework Consolidation” with regard to the data collection, data provenance, data safeguarding, data curation, data linking, data analytics and data sharing methods that are applicable to ICARUS. D3.3 provides the updated design and specifications of the services of the ICARUS platform, as well as of the components involved in them, to WP4 “Platform Agile Development and Deployment” that delivers the implementation of these services following the approach formulated in the WP3 activities. Finally, the feedback that will be collected from the continuous evaluation of the platform as a result of the WP5 (ICARUS Data Value Chain Demonstration) activities is constantly fed in WP3 and will drive the updates and adjustments in both the design and specifications of the services of the platform, as well as the overall integrated ICARUS platform and its components in the future documentation of the WP3 results (namely, in D3.4 and D3.5).

1.4 Structure

The structure of the document is as follows:

- In Section 2, the complementary documentation of the architecture of the integrated ICARUS platform is presented. Within this context, each component of the architecture is presented, focusing on the positioning of the component within the architecture, the platform functionalities that the component is involved in and the interactions of the component with the rest of the components for the implementation of these functionalities.
- In Section 3, the updated documentation of the design and specifications of the ICARUS platform's components is presented. In this section, for each component an overview containing the core functionalities of the component is presented. Furthermore, the involvement of the component in the ICARUS services and the designed ICARUS platform's workflows is outlined focusing on the interactions with the rest of the components and the interfaces offered by the component. Finally, the technical details of the interfaces provided by each component is documented.
- Section 4 concludes the deliverable, outlining the main findings of the deliverable which will guide the development efforts of the consortium.

2 ICARUS Platform Architecture

Within the context of deliverable D3.1, the high-level architecture of the ICARUS platform has been presented as the outcome of the thorough analysis of the technical requirements documented in section 4 that were later translated into technological, beyond the state of the art, software modules whose implementation is performed within the context of WP4.

The ICARUS architecture is a modular architecture that was designed with a high level of flexibility and adaptability. The main principle of the ICARUS architecture is to support the smooth and effective integration of all the designed software modules in which multiple technologies and tools are utilised towards the realisation of the designed workflows that will enable the aim of the ICARUS to deliver a novel big data platform for the aviation data value chain. The key aspects of the ICARUS architecture is the functional decomposition and the strict separation of concerns, as well as the dependencies identification and the design of a complete data flow. Within this architecture, each component is designed with the aim of delivering specific business services with a clear context, scope and set of features. The complete data flow is described in the form of interactions between the designed components that will enable the effective implementation of the designed workflows. Furthermore, the ICARUS architecture enables the interoperability of the various components that facilitate the execution of big data analytics and sharing of data through secure, transparent and advanced functionalities and features. To ensure this, all components of the ICARUS architecture provide well-defined interfaces to enable the seamless integration and operation of the integrated platform.

The presented high-level architecture of the platform depicts the entire lifecycle of the ICARUS platform that spans from the data preparation and data collection, the data exploration, the asset brokerage and data recommendation, to the data analysis and visualization, supplemented with advanced data security on all steps of the lifecycle. Furthermore, the lifecycle of the platform is complemented by added value steps such as notifications and usage analytics.

This high-level architecture had driven the implementation and release of the Alpha Version of the ICARUS platform, that was presented in D4.1, as well as the Beta version of the platform that is presented in D4.2. Figure 2-1 illustrates the high-level architecture of the ICARUS platform, along with the relevant information for the technologies and tools exploited by each component. The high-level architecture remained unaffected in terms of design, functionalities and interactions, as no additional requirements were identified requiring the introduction of any adjustment or refinement.

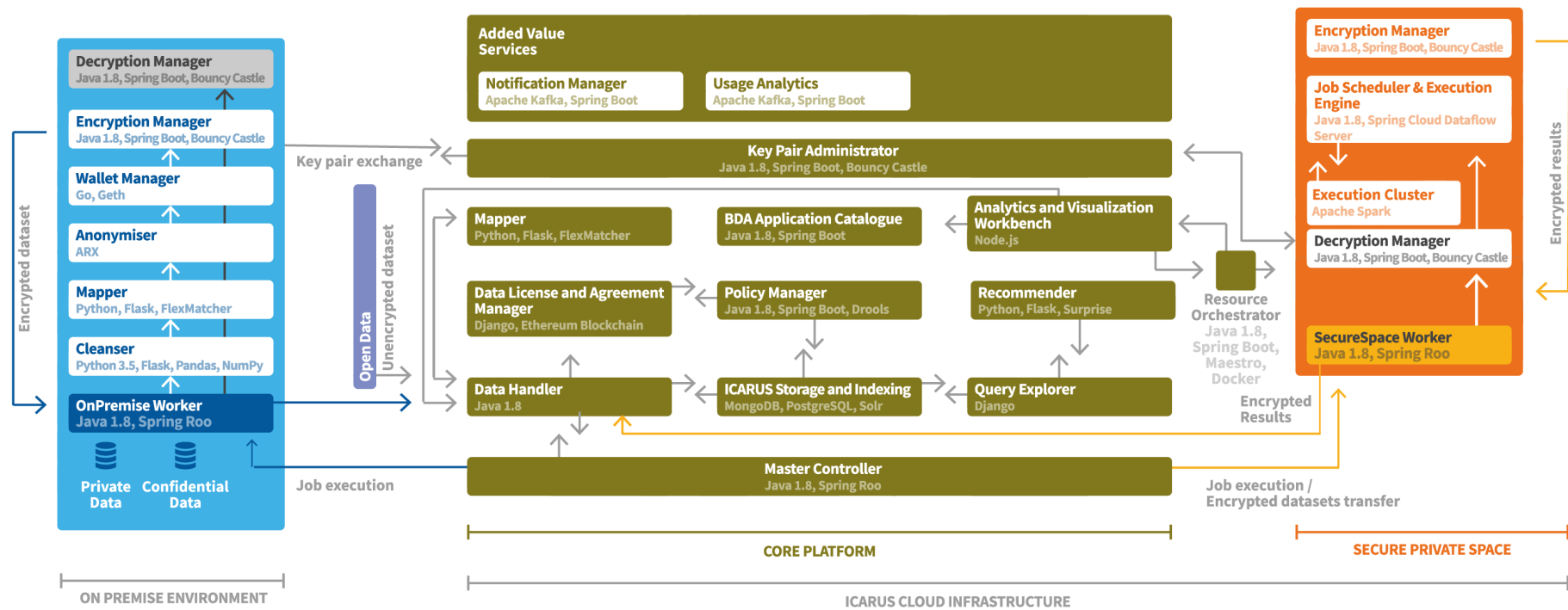


Figure 2-1: ICARUS high-level architecture

The ICARUS architecture is conceptually divided in three main tiers, the **On Premise Environment**, the **Core ICARUS platform** and the **Secure and Private Space**. Each tier is undertaking a set of functionalities of the ICARUS platform depending on the execution environment and context.

The **On Premise Environment** is composed by multiple components running on the data provider's environment with the main purpose to prepare the data provider's private or confidential datasets in order to be uploaded in the ICARUS platform. To facilitate the data preparation, the Master / Worker paradigm is utilised. The **OnPremise Worker** running on the On Premise Environment provides the interface to the Master Controller running on the Core ICARUS platform in order to receive the set of instructions for the data preparation or the local download of a dataset and its decryption. These instructions are interpreted and translated into a set of tasks that the OnPremise Worker is responsible to execute locally utilising the set of components running on the On Premise Environment for each specific task.

The **Cleanser** component provides the data cleansing functionalities of the platform according to the instructions provided by the OnPremise Worker through its relative interface. Based on the received instructions, the Cleanser performs a set of techniques for simple and more advanced cleansing operations over datasets that contain erroneous or "dirty" data by detecting or correcting corrupted, incomplete, incorrect and inaccurate records from datasets with a variety of rules and constraints. The **Mapper** component is responsible for the configuration and execution of the relative mapping instructions in collaboration with the OnPremise Worker. As the Mapper practically undertakes the harmonisation process of the dataset, upon receiving the mapping instructions, it performs the mapping of the fields of the dataset to the ICARUS common aviation model in a semi-automatic way, ensuring that the dataset will conform to the ICARUS schema when uploaded to the platform. The **Anonymiser** component is providing the data anonymisation functionalities which include filtering or hiding the private, sensitive or personal data that cannot be disclosed outside the data provider's premises, corporate network or personal filesystem by applying a set of anonymisation techniques in order to deal with privacy issues and the protection of sensitive information. As with the rest of the components of the On Premise Environment, the Anonymiser is providing the interface for the receival of the anonymisation instructions from the OnPremise Worker. The **Wallet Manager** is facilitating the operation of the Data License and Agreement Manager by providing a set of functionalities such as the generation and management of the blockchain account of the user and the interaction with the blockchain for the smart contract signature process. The **Encryption Manager** is undertaking all essential encryption processes with regard to encryption of the data provider's dataset. The Encryption Manager receives the relevant encryption instructions from the OnPremise Worker via the

provided interface and performs the encryption of the data provider's dataset with the encryption cipher mechanism that it provides. Furthermore, the Encryption Manager facilitates the dataset sharing, upon the agreement of the data provider and the data consumer, with the generation of the appropriate decryption keys and the secure transmission of the corresponding decryption key from the data provider to the data consumer by interacting with the Decryption Manager and the Key-Pair Administrator. The **Decryption Manager** is enabling the decryption of the dataset on the On Premise Environment when an encrypted dataset is downloaded locally as per the instructions provided by the OnPremise Worker, provided that a valid smart contract exists permitting the downloading of the specific dataset locally. The Decryption Manager provides the mechanisms to verify the identity of the data consumer via a certificate or public key, to request for the decryption key from the data provider and the decryption mechanism in order to temporarily reproduce the encryption key in order to decrypt the dataset.

The **Core ICARUS platform** is composed by multiple interconnected components running on the ICARUS infrastructure. The **Master Controller** is responsible for compiling and providing a set of instructions to be executed by the OnPremise Worker and the SecureSpace Worker following the Master / Worker paradigm as explained also above. The Master Controller is providing the interface that is utilised from the rest of the components of the Core ICARUS platform in order to submit the set of instructions that will be sent to the corresponding workers running on the On Premise Environment and the Secure and Private Space in order to be executed.

The **Data Handler** undertakes the role of the data gateway in the ICARUS architecture enabling the functionalities related to the availability of the data assets and their respective metadata in the platform. Through the interfaces offered by the Data Handler and the interactions with the relative workers through the Master Controller, it enables the uploading of the produced data provider's private or confidential dataset or the data generated as a result of a data analysis, as well as the downloading of datasets from the platform to the end user's On Premise Environment and/or to a Secure and Private Space. Furthermore, the Data Handler is handling the whole lifecycle management of the metadata of the data assets, while also providing a layer above the ICARUS storage, providing services and interfaces for storing or retrieving information from it that are exploited by the rest of the components of the Core ICARUS platform. The **Mapper** instance running on the Core ICARUS platform is directly interacting with the Data Handler through a set of interfaces with a dual purpose: (a) to perform the harmonisation process of the datasets originating from open data sources, and (b) to manage the semi-automated mapping of a dataset that is checked in in the ICARUS platform, to the ICARUS common aviation data model. The **Data License and Agreement**

Manager is responsible for the asset brokerage functionalities of the platform, providing the processes and interfaces that enable the preparation, drafting, signing and activation, as well as the enforcement of smart contracts that represent the data sharing agreements between platform users. The **Key Pair Administrator** is performing the signalling operations for the exchange of the decryption key between the data provider and the data consumer. It acts as the mediator between the Decryption Manager residing at the data consumer side and the Encryption Manager residing at the data provider side in order to establish a secure SSL-enabled connection for the exchange of the decryption key. The **Policy Manager** is implementing the sophisticated access control mechanism of the ICARUS platform that is based on the ABAC model and XACML standard. The Policy Manager performs the user management process, the complete access policy lifecycle management by interacting with the Data Handler through a set of interfaces and the access policy enforcement that controls and regulates the access of any resource via the dedicated interfaces, taking into account whether there is an active data contract by interacting with the Data License and Agreement Manager. The **ICARUS Storage and Indexing** component is providing the storage capabilities of the platform facilitating the data access and storage operations with two storage solutions, namely MongoDB and PostgreSQL, as well as the indexing capabilities over multiple complex datasets with the flexible and efficient search engine as provided by Solr. The **Query Explorer** encapsulates the intuitive environment that facilitates a data query definition with enhanced functionalities such as dynamic field selection and filter definition. The Query Explorer offers advanced dataset exploration and discoverability functionalities by interacting with the Solr search engine, while also interacting with the Policy Manager through the respective interface in order to ensure that the appropriate access policy filter is applied on the formulated query. Finally, it interacts with the Recommender component in order to retrieve and display the proper data recommendation to the users. The **Recommender** is providing the enhanced recommendation functionalities that enable the dataset exploration and discoverability. As such, the Recommender interacts with the Data Handler in order to obtain the required information from the ICARUS Storage and with the Query Explorer in order to provide the recommendations and suggestions for additional related datasets that can be explored or utilised during the search and query process.

The **Analytics and Visualisation Workbench** is providing the environment: (a) where the users of the platform are able to design, execute and monitor the data analytics workflows and (b) where the visualisation and dashboards are displayed. The Analytics and Visualisation Workbench is enabling the design of an ICARUS application in which the user is able to: (a) select an algorithm from the extended list of supported algorithms and set the corresponding parameters according to his/her needs, (b) select the datasets for the analysis from the list of

datasets the user owns or has legitimate access, and (c) select the visualisations from the variety of visualisations and dashboards that the platform is offering, and store it in the BDA Application Catalogue. While the design of the ICARUS application is performed in the Analytics and Visualisation Workbench, the execution of this application and the underlying data analysis is performed within the Secure and Private Space. To achieve this, the Analytics and Visualisation Workbench is interacting through the corresponding interfaces with the Resource Orchestrator for the provisioning of the Secure and Private Space, the Data Handler for the transferring of the datasets to the Secure and Private Space and the produced results from the Secure and Private Space to the ICARUS Storage with the help of the Master Controller, and finally with the Job Scheduler and Execution Engine for the actual execution of the designed application. The **BDA Application Catalogue** implements a repository of the ICARUS applications created by the users of the platform. As such, the ICARUS applications can be stored, retrieved, modified and loaded in the Analytics and Visualisation Workbench by the users at any time. The purpose of the BDA Application Catalogue is to enable the reuse of the designed data analytics workflows from the users, as well as the sharing of these workflows among the users through a defined license in order to empower the analytical capabilities of the platform. For this purpose, the Analytics and Visualisation Workbench is interacting with BDA Application Catalogue by a set of interfaces that are provided by the latter.

In addition to the aforementioned components, the ICARUS platform is supported by supplementary components with the aim of providing added-value services to the users of the platform. The **Notifications Manager** is responsible for providing the updated information to the users with regards to the datasets or the scheduled analytics jobs following the publish-subscribe pattern. To achieve this, the components of the platform produce various events in a message queue provided by the Notifications Manager and the Notifications Manager provides the relevant notifications to the users. The **Usage Analytics** component is responsible for providing the tools that collect, analyse and visualise the usage of the various services and assets of the platform in order to extract useful insights and statistics. Following also the publish-subscribe pattern, a message queue is provided by the Usage Analytics and the corresponding components of the platform are pushing new events in the queue based on the observed activities. The Usage Analytics is providing meaningful platform utilization insights related to the usage of data and service assets, the usage statistics of the platform and the users' private space to both the users and the platform administrator.

The Resource Orchestrator is enabling the provisioning and management of the Secure and Private Spaces. To achieve this, the Resource Orchestrator is able to connect to the virtualised infrastructure in order to perform monitoring and management of the available resources, to

allocate and release the resources in the corresponding virtual machines, as well as deploy and manage the applications and services running on the virtual machines. The Resource Orchestrator is interacting with the Analytics and Visualisations Workbench via the interfaces it provides for receiving the deployment requests for the provisioning of the Secure and Private Space of a user, as well as for the deployment of the services that are operating in the Secure and Private Space. Additionally, the Resource Orchestrator is handling the requests from the Analytics and Visualisations Workbench for the stoppage / shutdown of the Secure and Private Space of a user.

As such, the **Secure and Private Space** is realised in the form of dedicated virtual machines that are spawned on demand so that each user is able to perform analysis in an isolated and secure environment. The Secure and Private Space contains a set of interconnected components that constitute the advanced analytics execution environment of the ICARUS platform. The **SecureSpace Worker** running on the Secure and Private Space is providing the interface for receiving a set of instructions from the Master Controller related to the transferring of the required datasets from the ICARUS Storage to the Secure and Private Space for the execution of an ICARUS application, as well as the transferring of the encrypted results of this execution back the Core ICARUS platform for storage and visualisation purposes. The SecureSpace Worker undertakes the responsibility of executing the received instructions with the use of the set of components that are running on the Secure and Private Space. The **Decryption Manager** running on the Secure and Private Space undertakes the responsibility to decrypt the datasets in order to be used in the data analysis or in the visualisation process. The Decryption Manager provides the interface to the SecureSpace Worker and the Job Scheduler and Execution Engine in order to receive the decryption instructions. As described above, the Decryption Manager performs the data consumer's identity verification, the request for the decryption key exchange and eventually the decryption of the encrypted dataset via the dedicated decryption mechanism on the Secure and Private Space. Once an analysis is triggered by the Analytics and Visualisation Workbench, the **Job Scheduler and Execution Engine** is responsible for the initiation and monitoring of the corresponding job and for providing the relevant status, as well as the analysis results, in the Analytics and Visualisation Workbench in order to be displayed to the users. The Jobs Scheduler and Execution Engine is interacting with the Analytics and Visualisation Workbench in order to receive the request for the execution of an ICARUS application and the Execution Cluster for the actual execution of the application. Additionally, it interacts with the Decryption Manager for decryption of the dataset that will be used and with the Encryption Manager for the encryption of the produced results.

As it is obvious from the description of the ICARUS architecture, for the realisation of the various workflows of the ICARUS platform, multiple interactions exist between the components of the platform in all three tiers. The description above briefly presented these interactions highlighting the need of effective exchange of information between the components. In the following section, the functionalities of each component, as well its interactions with the rest of the components towards the implementation of the ICARUS platform features is described in detail along with the technical details of the provided interfaces that facilitate the successful exchange of information.

It should be noted also that experimentation is currently performed on testbeds that are setup with the various available technologies, including the HDFS distributed filesystem, for the efficient data transfer between the Secure and Private Space and Core ICARUS platform, in order to opt for the best suitable option. The outcomes of this experimentation for the Secure and Private Space will further enrich the technology stack of the ICARUS platform and the changes that will be introduced will be documented in the upcoming deliverable, namely D3.4.

3 ICARUS Components Designs

3.1 Overview

The ICARUS architecture is designed in a modular manner and is composed of a set of key components that were designed each one with a distinct role and scope. To ensure that the ICARUS stakeholders' needs are addressed, the elicited technical requirements that express these needs were translated into concrete platform features that were grouped under the modular components of the platform.

In order to enable the efficient and effective design of the ICARUS services that will facilitate the implementation of the ICARUS platform offerings and features, the outcomes of WP1 and WP2, and the knowledge extracted from the work performed in WP3 were further analysed. As a result, 29 workflows were designed in total, as presented in deliverable D3.2, that depict the interactions of the users with the ICARUS platform, as well as the interactions of the various components of the platform. The analysis of these workflows produced the design specifications of 18 service bundles, that were also documented in deliverable D3.2, each one fulfilling a specific set of responsibilities and functionalities within the ICARUS platform making sure that all aspired ICARUS platform offerings and features are covered.

Each component of the ICARUS architecture is involved in at least one of these services, providing the set of functionalities that is required for the implementation of the service. From the analysis of the design specifications of the ICARUS services, as well as the analysis of the designed workflows, it is obvious that for the realisation of these workflows, the various services, as well as the underlying components that are involved in these services, are interacting towards the exchange of information through well-defined interfaces.

In the following subsections, the interactions of each component with the rest of the components of the ICARUS platform is presented, highlighting the requirements for the effective exchange of information and the provided interfaces that are offered in order to address these requirements. Furthermore, for each component the technical details of the interfaces that each component offers are documented.

3.2 Anonymiser

3.2.1 Services Outline

The Anonymiser component is responsible for providing a privacy and anonymisation toolset that enables the anonymisation functionalities of the ICARUS platform. In detail, the Anonymiser handles the various privacy concerns and limitations of the data provider's dataset, as well as the protection of commercially sensitive, private or personal information that is incorporated in this dataset.

As described in deliverable D3.2, the Anonymiser component is involved in the Data Anonymisation service that is included in the Data Preparation workflow and is implementing the data anonymisation process, a generic process that is highly customisable based on the data provider's needs. Besides the dataset that the anonymisation process will be applied on, the data provider is responsible for providing the required configuration to the Anonymiser in order for the Anonymiser to customise the data anonymisation process based on received configuration. This configuration is provided in the form of anonymisation rules that are formulated based on his/her expertise over the content of the dataset, in which the anonymisation models, as well as the anonymisation technique and the anonymisation level from the selected model, are set, either at a field-level or at a dataset-level. The Anonymiser is responsible for the interpretation of these rules and the execution of the customised, based on these rules, data anonymisation process.

The Anonymiser component resides on the On Premise Environment and interacts only with the OnPremise Worker in order to receive the instructions containing the dataset's access information and the anonymisation rules. These instructions are formulated as one of the steps of the Data Preparation workflow and are initially provided to the Master Controller before they are dispatched to the Anonymiser through the OnPremise Worker. Hence, the Anonymiser component provides the interface that receives the formulated instructions from the OnPremise Worker. Once the instructions are received, the Anonymiser initiates the customised data anonymisation process over the selected dataset and an acknowledgement is provided to the OnPremise Worker. During this process execution, the OnPremise Worker can retrieve the current status of the ongoing data anonymisation process execution through a dedicated endpoint that is provided by the Anonymiser. Once the execution is finalised, the produced anonymised dataset is saved locally on the On Premise Environment and the Anonymiser informs the OnPremise Worker for the completion of the process via the respective endpoint that is provided by the OnPremise Worker.

It should be mentioned that the upcoming versions of the Anonymiser will include the re-identification risk assessment, as described in the ICARUS Anonymisation method of deliverable D2.3, and the endpoint that will provide the results of the risk assessment in order to be obtained by the data provider.

The details of the interfaces described above are presented in the following sub-section.

3.2.2 Interfaces

In the following tables the interfaces of the Anonymiser are defined.

Table 3-1: Anonymiser - initiate process

| |
|-----------------------------------|
| ICARUS Technical Interface |
|-----------------------------------|

| | |
|-------------------------------|---|
| Technical Interface ID | AN01 |
| Endpoint Name | Initiate data anonymisation process |
| Endpoint Description | Receives the data anonymisation instructions by the OnPremise Worker as set by the data provider and initiates the customised data anonymisation process based on these instructions |
| Component | Anonymiser |
| Endpoint URL | https://hostname[:port]/v1/anonymiser/instructions |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "processId": "string", "filePath": "string", "type": "anonymisation" "rules": [{"original_field": "string", "icarus_field": "string", "technique": "string", "method": "string", "level": "string"}] }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-2: Anonymiser - process status

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | AN02 |
| Endpoint Name | Ongoing data anonymisation process status |
| Endpoint Description | Receives the relevant anonymisation process id and returns the current execution status of the process. |
| Component | Anonymiser |
| Endpoint URL | https://hostname[:port]/v1/anonymiser/process-status/{processId} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • processId: The id of the process. |
| Request Body | None |
| Response Body | <pre>{ "processId": "string", "currentStatus": "NOTSTARTED INPROGRESS COMPLETED FAILED" }</pre> |

3.3 Cleanser

3.3.1 Services Outline

The Cleanser component is undertaking the responsibility of providing the data cleansing functionalities of the ICARUS platform. Within this context, the Cleanser provides the necessary data cleansing mechanisms that correct, clean and complete the provided dataset towards the maximization of the accuracy, completeness, correctness and usability of the dataset and the minimization of their data quality, reliability and integrity issues.

As described in deliverable D3.2, the Cleanser component is involved in the Data Cleansing service that is part of the Data Preparation workflow. In detail, the Cleanser component is responsible for the implementation of the data cleansing process that is provided by the Data Cleansing service. The data cleansing process incorporates a set of sequentially executed steps that include: (a) the data preliminary analysis, (b) the data validation, (c) the data cleansing, (d) the data completion and (e) the cleansing assessment sub-processes. All of these sub-processes are executed internally by the Cleanser component and their execution is based on a set of data validation, data cleansing and data completion rules that are defined by the data provider according to his/her needs. The rules are defined on a field-level that contain the list of constraints for a specific field and the corrective or removal action, in terms of cleansing or missing value handling, that will be performed in the case of a conformance error. In this way, the data cleansing process is tailored based on the nature of the dataset, as well as the requirements that are set by the data provider. The Cleanser is responsible to process these rules and apply them on top of the selected dataset in order to produce an accurate, complete and correct dataset that will be effectively used in high-quality data analysis.

The Cleanser component is one of the components that are residing on the On Premise Environment and undertakes the task of executing the instructions related to data cleansing as provided by the OnPremise Worker. Hence, the Cleanser component interacts only with the OnPremise Worker via a dedicated endpoint in order to receive these instructions that contain the dataset's access information and the corresponding data validation, data cleansing and data completion rules. These rules are defined within the context of the Data Preparation workflow and are provided to Cleanser component directly through the OnPremise Worker that initially collects them from the Master Controller running on the Core ICARUS platform. Once the Cleanser receives the instructions, it initiates the data cleansing process and the OnPremise Worker can obtain the current execution status of the process via a dedicated endpoint provided by the Cleanser. Finally, once the Cleanser has finished the

data cleansing process, the cleansed dataset is saved locally on the On Premise Environment and the Cleanser informs the OnPremise Worker for the process completion via the respective endpoint that is provided by the OnPremise Worker. Additionally, the Cleanser provides an interface from which the OnPremise Worker can obtain the log records that contain all the identified errors, the corrective or removal actions taken during the execution of the cleansing workflow for a specific job.

The details of the interfaces described above are presented in the following sub-section.

3.3.2 Interfaces

In the following tables the interfaces of the Cleanser are defined.

Table 3-3: Cleanser - initiate process

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | CL01 |
| Endpoint Name | Initiate data cleansing process |
| Endpoint Description | Receives the data cleansing instructions by the OnPremise Worker as configured by the data provider and the path of the dataset and initiates the data cleansing process by applying these rules on the selected dataset |
| Component | Cleanser |
| Endpoint URL | https://hostname[:port]/v1/cleanser/instructions |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "processId": "string", "filePath": "string", "type": "cleaning", "rules": { "cleaning": [{"original_field": "string", "icarus_field": "string", "rule": "string", "title": "string", "arguments": "string"}], "validation": [{"original_field": "string", "icarus_field": "string", "rule": "string", "title": "string", "arguments": "string"}], "missing": [{"original_field": "string", "icarus_field": "string", "rule": "string", "title": "string", "arguments": "string"}] } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-4: Cleanser - process status

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | CL02 |
| Endpoint Name | Ongoing data cleansing process status |
| Endpoint Description | Receives the relevant cleansing process id and returns the current execution status of the process. |
| Component | Cleanser |
| Endpoint URL | https://hostname[:port]/v1/cleanser/process-status/{processId} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> processId: The id of the process. |
| Request Body | None |
| Response Body | <pre>{ "processId": "string", "currentStatus": "NOTSTARTED INPROGRESS COMPLETED FAILED" }</pre> |

Table 3-5: Cleanser - obtain log records

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | CL03 |
| Endpoint Name | Obtain the log records of a completed cleansing process |
| Endpoint Description | Receives the relevant cleansing process id and return the log record of the completed cleansing process. |
| Component | Cleanser |
| Endpoint URL | https://hostname[:port]/v1/cleanser/retrieve-records/{processId} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> processId: The id of the process. |
| Request Body | None |
| Response Body | <pre>{ "processId": "string", "records": [{"Violated Constraint": "string", "Correction Action": "string", "Attribute": "string", "original_field": "string", "icarus_field": "string"}] }</pre> |

3.4 Mapper

3.4.1 Services Outline

As documented in deliverables D3.1 and D3.2, the Mapper is the component that undertakes all tasks pertaining to the mapping of the dataset's fields to the ICARUS common data model, both for datasets owned and provided by ICARUS stakeholders and for the ones coming from open data sources. The identification of such links and the generation of mappings between each dataset's schema and the ICARUS model is imperative to enable the provision of numerous ICARUS functionalities. Specifically, as already explained in the aforementioned deliverables, various ICARUS components and services (including dataset search and exploration, data integration and analysis, etc.) rely on the fact that datasets in ICARUS conform to a unique data model. Therefore, the Mapper is an important part of the Data Preparation workflow and each of its functionalities corresponds to one of the services that compose the "Data Mapping service", as presented in deliverable D3.2.

The Mapper has a dual presence in the ICARUS architecture, in the sense that some functionalities are provided inside the core ICARUS platform, whereas some limited functionality is implemented in the context of the On Premise Environment, the latter being similar to the way other components participating in the Data Preparation workflow operate, such as the Anonymiser and the Cleanser that were described in the previous sections of the current deliverable. This differentiating point constitutes a design decision partially explained by the fact that the mapping process needs to be executed also for the open data being imported in the platform - where no On Premise Environment is available. Moreover, it increases efficiency, as the need to transfer the latest common ICARUS data model locally in order to perform the mapping is eliminated.

The part of the Mapper that resides in the core platform interacts with the Data Handler in order to perform the following tasks: (1) generate the mapping for a dataset, (2) store an uncompleted mapping, which is required when a user wants to save mapping instructions either created exclusively by the Mapper or edited manually without being certain that these correspond to the final mapping, i.e. without validating that the mapping should be applied on the underlying data, (3) retrieve a mapping (marked as completed or not), (4) allow the user to edit an uncompleted mapping and (5) store a completed mapping, i.e. by validating that the included mapping instructions can be applied on the dataset by the Mapper services that reside in the On Premise Environment. It should be mentioned that the ICARUS Mapper is a self-learning component, as it leverages every new validated (completed) mapping in order to re-train and enhance the underlying mapping algorithm. This means that when task (5) in the above list is performed, the service being invoked also invokes the algorithm training

service. The creation of the mapping instructions constitutes a step of the Data Preparation workflow, but in order for the instructions to be executed, all data preparation instructions need to be provided, as parts of a data check-in job, which will be discussed in Section 3.9. Once all instructions are in place, the Data Handler, the Master Controller and the On Premise Worker interact (through services described in the corresponding sections), until the complete set of instructions has been made available to the Worker. Then, the Worker will interact with the On Premise part of the Mapper in order to execute the mapping instructions and ensure that the dataset will conform to the ICARUS schema when uploaded to the platform.

The interfaces that have been implemented to provide the above functionalities are documented in the following sub-section.

It should be mentioned that future versions of the Mapper will include the data model lifecycle management and evolution services required to ensure that the ICARUS data model can adapt to the evolving needs of the aviation industry and that all relevant/interesting datasets in this context can be handled by the platform.

3.4.2 Interfaces

The Mapper interfaces are documented in the following tables.

Table 3-6: Mapper - calculate mapping

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | MAP01 |
| Endpoint Name | Calculate Mapping |
| Endpoint Description | Used for calculating the mapping between a dataset uploaded by a user and the ICARUS data model |
| Component | Mapper |
| Endpoint URL | https://icarus_platform[:port]/api/v1/mapper/calculate-mapping |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> dcid: The id of the data checkin job |
| Request Body | N/A |
| Response Body | <pre>{ "icarus_fields": [{ "value": "aircraftModel", "definition": " An aircraft model represents a generic specification that describes the characteristics of a specific type of aircraft that has been manufactured." }, { "value": "ICAOAircraftType",</pre> |

| | |
|--|---|
| | <pre> "definition": " An aircraft type designator is a two-, three- or four-character alphanumeric code designating every aircraft type (and some sub-types) that may appear in flight planning as defined by the International Civil Aviation Organization." },], "mapping_records": [{ "field_type": "string", "icarus_field": "IATAAircraftType", "index": 0, "original_field": "ACT3" }, { "field_type": "datetime", "icarus_field": "actualDepartureTime", "index": 1, "original_field": "ACT5" }] } </pre> |
|--|---|

Table 3-7: Mapper - save mapping

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | MAP02 |
| Endpoint Name | Save Mapping |
| Endpoint Description | Used for storing an unfinished mapping in the database (i.e. when the user saves a mapping without it being complete) |
| Component | Mapper |
| Endpoint URL | https://icarus_platform[:port]/api/v1/mapper/save-model |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre> { "mapping": [{ "field_type": "string", "icarus_field": "IATAAircraftType", "index": 0, "original_field": "ACT3" }, { "field_type": "datetime", "icarus_field": "actualDepartureGateCloseTime", "index": 1, "original_field": "ACT5" }], "dcid": "1" } </pre> |

| | |
|----------------------|---|
| | } |
| Response Body | “Success” when the mapping was stored successfully or an error message indicating the problem otherwise |

Table 3-8: Mapper - save mapping and train model

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | MAP03 |
| Endpoint Name | Save Mapping and Train Model |
| Endpoint Description | Used for storing the mapping in the database when it is completed and for training the ICARUS data model with the newly created mapping |
| Component | Mapper |
| Endpoint URL | https://icarus_platform[:port]/api/v1/mapper/train-and-save-model |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "mapping": [{ "field_type": "string", "icarus_field": "IATAAircraftType", "index": 0, "original_field": "ACT3" }, { "field_type": "datetime", "icarus_field": "actualDepartureGateCloseTime", "index": 1, "original_field": "ACT5" }], "dcid": "1" }</pre> |
| Response Body | “Success” when the mapping was stored successfully or an error message indicating the problem otherwise |

Table 3-9: Mapper - initiate mapping process

| ICARUS Technical Interface | |
|-------------------------------|-------------------------------|
| Technical Interface ID | MAP04 |
| Endpoint Name | Initiate data mapping process |

| | |
|-----------------------------|--|
| Endpoint Description | Receives the mapping instructions by the OnPremise Worker as configured by the data provider and the path of the dataset and applies them on the selected dataset |
| Component | Mapper |
| Endpoint URL | https://hostname[:port]/v1/mapper/instructions |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "processId": "string", "filePath": "string", "type": "mapping", "rules": [{ "icarus_field": "IATAAircraftType", "index": 0, "original_field": "ACT3" }, { "icarus_field": "actualDepartureGateCloseTime", "index": 1, "original_field": "ACT5" }] }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-10: Mapper - process status

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | MAP05 |
| Endpoint Name | Ongoing data mapping process status |
| Endpoint Description | Receives the relevant mapping process id and returns the current execution status of the process. |
| Component | Mapper |
| Endpoint URL | https://hostname[:port]/v1/mapper/process-status/{processId} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • processId: The id of the process. |
| Request Body | None |
| Response Body | <pre>{ "processId": "string", "currentStatus": "NOTSTARTED INPROGRESS COMPLETED FAILED" }</pre> |

3.5 Wallet Manager

3.5.1 Services Outline

The Wallet Manager is a supplementary component that handles all blockchain-related operations in the context of the On Premise Environment. It interacts only with the Blockchain nodes in order to query the status of a smart contract. This interaction is performed via Remote Procedure Calls (RPC); hence no interfaces are foreseen.

3.5.2 Interfaces

The Wallet Manager does not offer any interfaces.

3.6 Encryption Manager

3.6.1 Services Outline

The Encryption Manager undertakes the responsibility of ensuring the security and integrity of the data assets with the appropriate encryption mechanism and facilitating the secure and controlled sharing of encrypted datasets between the data provider and the data consumer.

The Encryption Manager is involved in the Data Encryption and Data Decryption service, as described in the deliverable D3.2, along with the Decryption Manager and the Key Pair Administrator. In detail, the Encryption Manager component is responsible for: (a) providing the column-based encryption mechanism that utilises a symmetric key encryption for the security and integrity of the data assets, (b) supporting the SSL-enabled connection establishment with the Decryption Manager upon successful authorisation and identity verification, (c) providing the mechanism that generates the decryption symmetric key and securely transmitting it over the established SSL-enabled connection, and (d) providing the revocation process of a generated decryption key upon needs.

As part of the Data Preparation workflow described in deliverable D3.2, the Encryption Manager that resides on the On Premise Environment interacts with the OnPremise Worker in order to receive the encryption instructions that contain the columns of the respective dataset that must be encrypted based on the input of the data provider. Hence, the Encryption Manager provides the interface for receiving these instructions and once the instructions are received the encryption process is initiated. Furthermore, the Encryption Manager provides a dedicated endpoint from which the OnPremise Worker can obtain the current execution status of the encryption process. Once the encryption process is finished, the produced ciphertext is saved locally and the OnPremise Worker is informed that the process is completed via the endpoint that is provided by the OnPremise Worker. In the same manner, the Encryption Manager that resides on the Secure and Private Space interacts with the SecureSpace Worker and receives a request to encrypt the produced results of a data

analysis execution (in which all columns are encrypted), as depicted in the ICARUS application execution workflow of the Data Analytics and Visualisations workflows described in deliverable D3.2. For this purpose, an interface is provided by the Encryption Manager and the produced ciphertext is stored locally on the Secure and Private Space in order to be latter transmitted and stored in the ICARUS platform storage via the Data Handler.

As part of the data decryption workflow that is also described in deliverable D3.2 in the Data Security workflows, the Encryption Manager (e.g. of the data provider) is interacting with the Decryption Manager (e.g. of the data consumer) and the Key Pair administrator in order to enable the secure sharing of encrypted datasets. In particular, the Encryption Manager facilitates the data decryption process by providing an interface for receiving decryption requests. By interacting with the Key Pair Administrator and the Decryption Manager a secure connection is established, the proper decryption symmetric key is produced and is transmitted to the Decryption Manager. The decryption process is executed during the ICARUS application execution workflow that is presented in the Data Analytics and Visualisations workflows document in deliverable D3.2 and in the cases where data assets or the produced results of the analysis are downloaded locally or decrypted in order to be used in a visualisation with no differentiation.

It should be mentioned that the upcoming versions of the Encryption Manager will include the efficient revocation process for the generated symmetric decryption key with the support of the Key-Pair Administrator, in the case when access to an encrypted dataset must be revoked for a specific data consumer.

The details of the interfaces described above are presented in the following sub-section.

3.6.2 Interfaces

In the following tables the interfaces of the Encryption Manager are defined.

Table 3-11: Encryption Manager -initiate process

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | EM01 |
| Endpoint Name | Initiate data encryption process |
| Endpoint Description | Receives the instructions for the data encryption by the OnPremise Worker as set by the data provider and initiates the data encryption process based on these instructions. The endpoint is utilised also from the SecureSpace worker for the encryption of the data analysis results. |
| Component | Encryption Manager |
| Endpoint URL | https://hostname[:port]/v1/encryption/instructions |
| HTTP method | POST |

| | |
|---------------------------|--|
| Request Parameters | N/A |
| Request Body | <pre>{ "processId": "string", "filePath": "string", "type": "encryption", "instructions": [{"original_field": String, "icarus_field": "string"}] }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-12: Encryption Manager - process status

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | EM02 |
| Endpoint Name | Ongoing data encryption process status |
| Endpoint Description | Receives the relevant data encryption process id and returns the current execution status of the process. |
| Component | Encryption Manager |
| Endpoint URL | https://hostname[:port]/v1/encryption/process-status/{processId} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • processId: The id of the process. |
| Request Body | None |
| Response Body | <pre>{ "processId": "string", "currentStatus": "NOTSTARTED INPROGRESS COMPLETED FAILED" }</pre> |

Table 3-13: Encryption Manager - receive decryption request

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | EM03 |
| Endpoint Name | Receive data decryption requests |
| Endpoint Description | <p>Receives the data decryption request from the Key-Pair Administrator and initiates the data decryption process from the data provider's side. In the background, an attempt to establish an SSL-enabled connection is initiated and if it is established, the existence of a valid smart contract is confirmed and then the decryption symmetric key is transmitted over this connection.</p> <p>If the connection is successful, a positive response is sent to the Key-Pair Administrator else a negative response is sent</p> |

| | |
|---------------------------|---|
| Component | Encryption Manager |
| Endpoint URL | https://hostname[:port]/v1/encryption/init-decryption |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "data_asset_id": "string", "data_consumer_id": "string", "connection_details": {"connection_id": "string", "connection_URL": "string"} }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 401 Unauthorised |

3.7 Decryption Manager

3.7.1 Services Outline

The Decryption Manager component is responsible for enabling the secure and effective decryption of the encrypted datasets on the data consumer side when legitimate access has been obtained (based on a contract) without compromising the data privacy of the data provider.

The Decryption Manager is involved in the Data Encryption and Data Decryption service, as described in the deliverable D3.2, along with the Encryption Manager and the Key Pair Administrator. The Decryption Manager offers the decryption mechanism of the encrypted ciphertexts and facilitates the execution of the data decryption workflow that is described in deliverable D3.2 in the Data Security workflows. In detail, the Decryption Manager is involved in the process, along with the Encryption Manager and the Key Pair Administrator, in order to: (a) establish the required SSL-enabled connection with the Encryption Manager with the help of Key Pair Administrator and (b) acquire the produced from the Encryption Manager decryption symmetric key in order to perform the decryption of the ciphertext.

The Decryption Manager provides the interface that receives the requests for the decryption of the data assets from the rest of the components of the platform. Upon receiving the decryption request, the Decryption Manager initiates the decryption process by interacting with the Key Pair Administrator in order to establish the secure connection with the Encryption Manager and request for the decryption symmetric key. Provided that the request is accepted by the Encryption Manager after performing the necessary access validation processes, the Decryption Manager receives the produced decryption symmetric key and performs the decryption of the data asset. The produced encrypted data asset is saved locally in order to be exploited. As described also in section 3.6, the decryption process is performed

during the ICARUS application execution workflow, as presented in the Data Analytics and Visualisations workflows in deliverable D3.2, and during the download process or the visualisation process that is presented in the same section of D3.2.

The details of the interfaces described above are presented in following the sub-section.

3.7.2 Interfaces

In the following tables the interfaces of the Decryption Manager are defined.

Table 3-14: Decryption Manager - initiate process

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | DM01 |
| Endpoint Name | Initiate data decryption requests |
| Endpoint Description | Receives the data decryption request and initiates the data decryption process from the data consumer's side. In the background, the Key-Pair Administrator is contacted in order to setup the connection with corresponding data provider. Once the connection is set and the decryption symmetric key is received, the decryption process is performed. |
| Component | Decryption Manager |
| Endpoint URL | https://hostname[:port]/v1/decryption/initiate |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | { "data_asset_id": "string", "purpose": "DATAANALYSIS VISUALISE DOWNLOAD" } |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 401 Unauthorised |

Table 3-15: Decryption Manager - check connection response

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | DM02 |
| Endpoint Name | Check connection response |
| Endpoint Description | Process the connection response from the data provider. |
| Component | Decryption Manager |
| Endpoint URL | https://hostname[:port]/v1/decryption/connection-status |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | { |

| | |
|----------------------|--|
| | <pre>"data_asset_id": "string", "purpose": "DATAANALYSIS VISUALISE DOWNLOAD", "connection_id": "string", }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK with "connection_status": "CONNECTED" • 400 Bad Request with "connection_status": "REJECTED" |

Table 3-16: Decryption Manager -process decryption response

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DM03 |
| Endpoint Name | Process decryption response |
| Endpoint Description | Process the decryption response from the data provider and wait for the decryption symmetric key through the SSL-enabled connection. |
| Component | Decryption Manager |
| Endpoint URL | https://hostname[:port]/v1/decryption/response |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "data_asset_id": "string", "purpose": ["DATAANALYSIS", "VISUALISE", "DOWNLOAD"], "response_status": "APPROVED REJECTED" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request |

3.8 Key-Pair Administrator

3.8.1 Services Outline

The Key-Pair Administrator is the component complementing the data decryption process by enabling the establishment of the secure SSL-enabled connection between the Encryption Manager and the Decryption Manager in order to perform the secure exchange of the decryption symmetric key to perform the data decryption process.

The Key-Pair Administrator is involved in the Data Encryption and Decryption service together with the Encryption Manager and the Decryption Manager, as documented in deliverable D3.2. The scope of the Key-Pair Administrator is to perform all the signalling operations between the Decryption Manager residing at the data consumer side and the Encryption Manager residing at the data provider side. As such, the Key-Pair Administrator enables the

setup of the secure transport encryption by supporting the two-sided authorisation process and the identity verification.

As part of the data decryption workflow that is also described in deliverable D3.2 in the Data Security workflows, the Key-Pair Administrator provides the interface to receive the decryption request from the Decryption Manager and communicates with the Encryption Manager of the corresponding data provider in order to establish the secure connection. The appropriate SSL handshake is executed as a background operation executed between the Encryption Manager and the Decryption Manager. Once the connection is established, the Key-Pair Administrator is performing a data access request to the access policy enforcement endpoint of the Policy Manager, that is described in section 3.11, in order to verify that the data consumer is eligible to access to the requested dataset. Once the access is granted, the Key-Pair Administrator is informing the Encryption Manager for the access decision and the decryption symmetric key is generated and transmitted to the Decryption Manager through the secure connection.

It should be mentioned that the upcoming versions of the Key-Pair Administrator will include the support for the revocation process, that will be handled by the Encryption Manager, for the generated symmetric decryption key when access to an encrypted dataset must be revoked for a specific data consumer.

The details of the interfaces described above are presented in the following sub-section.

3.8.2 Interfaces

In the following tables the interfaces of the Key-Pair Administrator are defined.

Table 3-17: Key Pair Administrator - connection request

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | KPA01 |
| Endpoint Name | Receive connection request |
| Endpoint Description | Receives the connection request from the Decryption Manager running on the data consumer's side. It initiates the connection with the Encryption Manager running on the data provider's side. In the background, the Key-Pair Administrator discovers the connection details of the data provider and initiates a decryption request by passing the connection details of the data consumer to the data provider. If the connection details are accepted the two-sided authorisation process and the identity verification is performed and the connection is established. |
| Component | Key-Pair Administrator |
| Endpoint URL | https://hostname[:port]/v1/keypair-admin/connection-request |
| HTTP method | POST |

| | |
|---------------------------|---|
| Request Parameters | N/A |
| Request Body | <pre>{ "data_asset_id": "string", "data_consumer_id": "string", "connection_details": {"connection_id": "string", "connection_URL": "string"} }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 401 Unauthorised |

Table 3-18: Key Pair Administrator - check data access rights

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | KPA02 |
| Endpoint Name | Check data access rights |
| Endpoint Description | Receives a request to check the data assets rights and returns the data access verification. The Key-Pair Administrator acts as a mediator for the data access verification process that is performed by the Policy Manager. |
| Component | Key-Pair Administrator |
| Endpoint URL | https://hostname[:port]/v1/keypair-admin/access |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "data_asset_id": "string", "data_consumer_id": "string", "action": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 401 Unauthorised |

3.9 Data Handler

3.9.1 Services Outline

The Data Handler acts as a data gateway in the ICARUS architecture, as it is the component that provides the functionalities related to making data assets and their metadata available to and from the ICARUS platform, as well as among different platform components. Specifically, it supports the complete workflow of uploading proprietary and open datasets to

the platform, downloading datasets from the platform to the end user's On Premise Environment and/or to a Secure and Private Space, and uploading data generated in a Secure and Private Space back into the core platform's storage. Moreover, the Data Handler is responsible for the provision of the metadata of a dataset by its provider and is therefore the component that provides the corresponding interfaces. Data Handler, as shown in deliverable D3.2 and briefly explained above, holds an important role in the workflows of Data Preparation, and Data Collection and is also involved in the Notifications workflows in the case of notifications for new datasets. The Data Handler is therefore the core component involved in the metadata handling and the data upload services, also described in deliverable D3.2.

It should be noted that the component's functionality has been slightly extended compared to what was documented in deliverables D3.1 and D3.2 and now includes the definition of the dataset licensing and IPR metadata, previously seen as responsibility of the Data License and Agreement Manager component. This decision was made to ensure a consistent and smooth workflow regarding metadata provision by the dataset provider, as the Data Handler is the component responsible for the definition of all other dataset related metadata, both the ones automatically calculated and the ones manually provided by the data owners.

Furthermore, the Data Handler acts as a layer above the ICARUS storage, offering services for other components to store in and retrieve information from it. As expected by its wide scope, the component offers various endpoints which are documented in the following sub-section.

3.9.2 Interfaces

In the following tables the interfaces of the Data Handler are defined.

Table 3-19: Data Handler - create new data preparation job

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH01 |
| Endpoint Name | Create new data preparation (data checkin) job |
| Endpoint Description | Initiates the process of a new dataset check-in by creating a new data preparation job |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin |
| HTTP method | POST |
| Request Parameters | N/A |

| | |
|----------------------|--|
| Request Body | <pre>{ "complete": true, "completed": true, "configuration": { }, "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "data_asset_id": 0, "description": "string", "executionstate": "string", "name": "string", "user": { "department": "string", "firstname": "string", "id": 0, "lastname": "string", "organization": { "businessname": "string", "city": "string", "country": "string", "websiteurl": "string" }, "username": "string" }, "sample": { } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-20: Data Handler - get data preparation job

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH02 |
| Endpoint Name | Get data preparation (data checkin) job |
| Endpoint Description | Retrieves an existing data preparation job |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid} |
| HTTP method | GET |

| | |
|---------------------------|--|
| Request Parameters | <ul style="list-style-type: none"> dcid: The id of the data checkin job |
| Request Body | N/A |
| Response Body | <pre>{ "complete": true, "completed": true, "configuration": { }, "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "data_asset_id": 0, "description": "string", "executionstate": "string", "name": "string", "user": { "department": "string", "firstname": "string", "id": 0, "lastname": "string", "organization": { "businessname": "string", "city": "string", "country": "string", "websiteurl": "string" }, "username": "string" }, "sample": { } }</pre> |

Table 3-21: Data Handler - modify data preparation job

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH03 |
| Endpoint Name | Modify data preparation (data checkin) job |
| Endpoint Description | Initiates the process of a new dataset check-in by creating a new data preparation job |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> dcid: The id of the data checkin job |

| | |
|----------------------|---|
| Request Body | <pre> { "complete": true, "completed": true, "configuration": { }, "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "data_asset_id": 0, "description": "string", "executionstate": "string", "name": "string", "user": { "department": "string", "firstname": "string", "id": 0, "lastname": "string", "organization": { "businessname": "string", "city": "string", "country": "string", "websiteurl": "string" }, "username": "string" }, "sample": { } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-22: Data Handler - create data preparation instructions

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | DH04 |
| Endpoint Name | Create data preparation instructions |
| Endpoint Description | Accepts and stores a set of data preparation instructions for the specified step of the process |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid}/{step} |
| HTTP method | PUT |

| | |
|---------------------------|---|
| Request Parameters | <ul style="list-style-type: none"> • step: step of the data preparation process (datacheckin job) • dcid: id of the data preparation job |
| Request Body | <pre>{ "completed": true, "context": { }, "datacheckinjobid": 0, "id": 0, "updated": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-23: Data Handler - get data preparation instructions

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH05 |
| Endpoint Name | Get data preparation instructions for a specific job for a specific step |
| Endpoint Description | Retrieve a set of data preparation instructions for the specified step of the specified data preparation job |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid}/{step} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • step: step of the data preparation process (datacheckin job) • dcid: id of the data preparation job |
| Request Body | N/A |
| Response Body | <pre>{ "completed": true, "context": { }, "datacheckinjobid": 0, "id": 0, "updated": { "date": 0, "day": 0, "hours": 0,</pre> |

| | |
|--|--|
| | <pre> "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 } } </pre> |
|--|--|

Table 3-24: Data Handler - execute data preparation instructions

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | DH06 |
| Endpoint Name | Execute data preparation instructions |
| Endpoint Description | When this service is invoked, the Data Handler will assemble all data preparation instructions of the specified datacheckin job, re-structure them and send them to the Master Controller in order to initiate their execution process |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid}/execute |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: data preparation (datacheckin) job id |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-25: Data Handler - get data preparation status

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | DH07 |
| Endpoint Name | Get data preparation status |
| Endpoint Description | Returns the status of the specified data preparation job |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid}/status |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> dcid: the data preparation job id |
| Request Body | N/A |

| | |
|----------------------|---|
| Response Body | Returns the status of the job: <ul style="list-style-type: none"> • Uncompleted • InProgress • MappingState • CleaningState • EncryptionState • AnonymisationState • IndexingState • Completed • Failed Else an error message. |
|----------------------|---|

Table 3-26: Data Handler - upload data sample

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | DH08 |
| Endpoint Name | Upload data sample |
| Endpoint Description | Handles the upload of a dataset sample in the ICARUS platform |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/datacheckin/{dcid}/sample |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> • dcid: the data preparation job id |
| Request Body | <pre>{ "dataname": "string", "fielddata": "string", "fielddatacleaned": "string", "id": "string", "user": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-27: Data Handler - upload data

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | DH09 |
| Endpoint Name | Upload data |
| Endpoint Description | Handles the upload of a dataset in the ICARUS platform |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/upload/{dcid}/dataset |

| | |
|---------------------------|---|
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> • dcid: the data preparation job id |
| Request Body | Header: <i>Content-Type: multipart/form-data;</i> <pre>{ "dataname": "string", "id": "string", "user": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-28: Data Handler - download data

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH10 |
| Endpoint Name | Download data |
| Endpoint Description | Initiates the process for downloading a dataset |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/download/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • id: the dataset id |
| Request Body | N/A |
| Response Body | <pre>{ "dataname": "string" }</pre> |

Table 3-29: Data Handler - transfer data

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH11 |
| Endpoint Name | Transfer data |
| Endpoint Description | Initiates the process for transferring the specified dataset from the core ICARUS storage to a Secure and Private space. The actual file transfer is handled internally with the appropriate backend process and the relevant driver for the DB. |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/download/{id} |
| HTTP method | POST |

| | |
|---------------------------|--|
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-30: Data Handler - add metadata

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | DH12 |
| Endpoint Name | Add metadata to the specified dataset |
| Endpoint Description | Add metadata fields to the specified dataset |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/data-asset/{id}/metadata |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | <pre>{ "metadata": ["core": {}, "semantic": {}, "distribution": {}, "sharing": {}, "preservation": {}] }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 201 Created 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-31: Data Handler - get metadata

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DH13 |
| Endpoint Name | Get dataset metadata |
| Endpoint Description | Get all metadata for the specified dataset |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/data-asset/{id}/metadata |
| HTTP method | GET |

| | |
|---------------------------|---|
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | N/A |
| Response Body | <pre>{ "metadata": ["core": { }, "semantic": { }, "distribution": { }, "sharing": { }, "preservation": { }] }</pre> |

Table 3-32: Data Handler - update metadata

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | DH14 |
| Endpoint Name | Update dataset metadata |
| Endpoint Description | Update metadata fields of the specified dataset |
| Component | Data Handler |
| Endpoint URL | https://icarus_platform[:port]/api/v1/handler/data-asset/{id}/metadata |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | <pre>{ "metadata": ["core": { }, "semantic": { }, "distribution": { }, "sharing": { }, "preservation": { }] }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

3.10 Data License and Agreement Manager

3.10.1 Services Outline

In deliverables D3.1 and D3.2, the Data License and Agreement Manager was described as the component responsible for handling all processes related to the data licenses and IPR attributes, as well as the drafting, signing, and enforcing the smart data contracts that correspond to data sharing agreements between platform users. As such, the component,

which resides in the core ICARUS platform, participates in numerous of the workflows presented in deliverable D3.2 and plays a crucial role in the provision of data licensing and data brokerage services by ICARUS.

However, in order to ensure a smoother user experience and increased consistency, it was decided that the interface used for the definition of license related attributes of a given dataset should be part of the Data Handler, which also provides the interfaces for the definition of all other dataset metadata. The Data License and Agreement Manager is thus primarily responsible for the data brokerage functionalities and therefore interacts with various components in order to enable mainly the following processes:

A user (data consumer) should be able to issue a request to purchase a dataset owned by another user (data provider) and the request should be stored in the platform. After being notified of the request, the dataset provider should be able to refuse the purchase request for the dataset or initiate the data brokerage process by drafting a data contract with the appropriate terms. The drafted contract should then be uploaded in the Blockchain and the (prospective) data consumer, should be allowed to review it and accept it, reject it or propose changes accordingly. In case changes are proposed, the data provider should in turn be able to accept, reject or propose new changes. In order to perform these processes, a graphical interface is used by the data provider and the data consumer, but all changes are also propagated to the Blockchain by the Data License and Agreement manager, which updates the terms and/or the status of the smart contract accordingly. Once a contract is accepted by both parties, the component also allows the consumer to provide proof of payment, in which case the contract is validated. At all times, the component is responsible of checking and reporting on the status of a smart contract and of updating the status as needed. These functionalities are implemented through the interfaces reported in the following sub-section. It should be noted that the RPC commands used to implement the interaction with the Blockchain, although their functionality has been described, cannot be documented in the same format and are therefore not relevant to the information documented in the next sub-section.

3.10.2 Interfaces

In the following tables the interfaces of the Data License and Agreement Manager are defined.

Table 3-33: Data License and Agreement Manager - request to buy data asset

| ICARUS Technical Interface | |
|----------------------------|---------------------------|
| Technical Interface ID | DLAM01 |
| Endpoint Name | Request to Buy Data Asset |

| | |
|-----------------------------|--|
| Endpoint Description | Receives a request issued by an ICARUS user to purchase a data asset that is owned by another user |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/data-asset/buy-asset |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "asset_id": 0, "comments": "string", "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "data_consumer_id": 0, "data_owner_id": 0, "duration": 0, "fields": "string", "filters": "string", "id": 0, "request": true }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-34: Data License and Agreement Manager - retrieve request to buy data asset

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM02 |
| Endpoint Name | Retrieve a request to Buy a Data Asset |
| Endpoint Description | Retrieves the information related to a specific issued request to buy a data asset |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/data-asset/buy-asset/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • Id: the request id |
| Request Body | N/A |

| | |
|----------------------|--|
| Response Body | <pre> { "asset_id": 0, "comments": "string", "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "data_consumer_id": 0, "data_owner_id": 0, "duration": 0, "fields": "string", "filters": "string", "id": 0, "request": true } </pre> |
|----------------------|--|

Table 3-35: Data License and Agreement Manager - reject request to buy data asset

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM03 |
| Endpoint Name | Reject Request to Buy Data Asset |
| Endpoint Description | Changes the status of a buy data asset request to rejected |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/data-asset/buy-asset/{id}/reject |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the request to buy asset |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-36: Data License and Agreement Manager - accept request to buy data asset

| ICARUS Technical Interface | |
|-------------------------------|----------------------------------|
| Technical Interface ID | DLAM04 |
| Endpoint Name | Accept Request to Buy Data Asset |

| | |
|-----------------------------|--|
| Endpoint Description | Changes the status of a buy data asset request to accepted |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/data-asset/buy-asset/{id}/accept |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the request to buy asset |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-37: Data License and Agreement Manager - mark buy data asset as paid

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM05 |
| Endpoint Name | Mark Request to Buy Data Asset as Paid |
| Endpoint Description | Changes the status of a buy data asset request to paid (valid) |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/data-asset/buy-asset/{id}/paid |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the request to buy asset |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-38: Data License and Agreement Manager - create new contract

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM06 |
| Endpoint Name | Create new contract |
| Endpoint Description | Create a new smart contract for a data brokerage agreement |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/contract |
| HTTP method | POST |
| Request Parameters | N/A |

| | |
|----------------------|---|
| Request Body | <pre>{ "amount": 0, "hashed_asset_id": 0, "contract_address": "string", "currency": "string", "data_consumer_address": 0, "data_owner_address": 0, "duration": 0, "fields": [], "filters": [], "id": 0, "stage": "string", "tax": 0, "hashed_terms": [], "eth_address_manager": 0, "validated_at": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-39: Data License and Agreement Manager - get contract

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM07 |
| Endpoint Name | Get a contract |
| Endpoint Description | Get a specific smart contract for a data brokerage agreement |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/contract/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • id: the id of the contract |
| Request Body | N/A |

| | |
|----------------------|---|
| Response Body | <pre>{ "amount": 0, "hashed_asset_id": 0, "contract_address": "string", "currency": "string", "data_consumer_address": 0, "data_owner_address": 0, "duration": 0, "fields": [], "filters": [], "id": 0, "stage": "string", "tax": 0, "hashed_terms": [], "eth_address_manager": 0, "validated_at": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 } }</pre> |
|----------------------|---|

Table 3-40: Data License and Agreement Manager - update contract

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | DLAM08 |
| Endpoint Name | Update smart data contract |
| Endpoint Description | Update the terms and/or status (stage) of a smart contract for a data brokerage agreement |
| Component | Data License and Agreement Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/contract/{id} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the contract |
| Request Body | <pre>{ "amount": 0, "asset_id": 0, "contract_address": "string", "currency": "string", "data_consumer_id": 0, "data_owner_id": 0, "duration": 0, "fields": "string", "filters": "string", </pre> |

| | |
|----------------------|--|
| | <pre> "id": 0, "stage": "string", "tax": 0, "terms": [], "user_id": 0, "validated_at": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 } } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

3.11 Policy Manager

3.11.1 Services Outline

The Policy Manager is the component that is enabling the effective and robust protection of any type of resource of the ICARUS platform such as data, applications, any kind of system resources, as well as all other relevant objects. To achieve this, the Policy Manager offers a sophisticated authorisation engine that provides the reliable access control mechanism which enforces the selected restriction of access to these resources and eliminates the unauthorised disclosure to any of these resources.

As described in deliverable D3.2, the Policy Manager component is involved in the Policies Enforcement service that is involved in multiple workflows offering different functionalities on each one of them. The first core part of the Policy Manager component that facilitates the efficient and effective operation of the access control mechanism is the user management process, as presented in the Data Security workflows in section 2.7 of deliverable D3.2, having direct interaction the platform's user interface. Within this process, the creation (or registration), modification and suspension of the organisation entities, as well as the creation of the respective invitation link and token for the users (members) of each organisation that are invited by the organisation manager, is facilitated with a set of interfaces provided by the Policy Manager. Furthermore, the Policy Manager enables the registration of the invited users (members) and creation of their user profile, as well as the modification and suspension of

their user profile, with the required interfaces. Finally, the Policy Manager provides the respective interfaces for the user authentication via the corresponding login process.

The second core part of the Policy Manager component is related to the access policy lifecycle management, as presented also in the Data Security workflows in section 2.7 of deliverable D3.2, and includes the creation, update or deletion of an access control policy over a resource of the platform. This process is also involved in the manual metadata definition process described in the Data Collection workflows in section 2.3 and in the creation of the metadata of an ICARUS application as described in the Data Analytics workflows in section 2.4 in the deliverable D3.2. Hence, the Policy Manager provides the respective interface that handles all the operations for the access policy lifecycle management and interacts with the Data Handler component that dispatches the respective actions based on the input received by the user of the platform. These access policies are deployed on the authorisation engine in order to be used during the authorisation process.

The third core part of the Policy Manager component is responsible for the access policy enforcement that controls and regulates the access of any resource, as presented in the Data Security workflows in section 2.7 of deliverable D3.2, and includes the processing of the received access request, the evaluation of the relevant policies and required attributes and the formulation of an access control decision. This process is executed before any resource of the platform is accessed in order to safeguard the protection of the underlying resources. Furthermore, the access policy enforcement is involved in the datasets query and dataset exploration processes in order to formulate the access policy enforcement during the query and exploration step with an access control filter that is incorporated in the produced query to the underlying query engine. Finally, the access policy enforcement is involved in the ICARUS application design and the results visualisation workflows described in the Data Analytics workflows in section 2.4 in order to validate the access to the respective data assets during the data decryption process. To facilitate this process, the Policy Manager offers two options: (a) a specialised library that is integrated in the source of the component which requires the access policy enforcement functionalities in order to properly annotate all its interfaces so that the Policy Manager intercepts all requests in order to automatically formulate the access decision prior to the execution of the process offered by the respective interface, and (b) an access policy enforcement endpoint that is utilised on-demand by the rest of the ICARUS platform's components in all cases that access to an underlying resource is attempted.

The details of the interfaces described above are presented in the following sub-section.

3.11.2 Interfaces

In the following tables the interfaces of the Policy Manager are defined.

Table 3-41: Policy Manager - create organisation

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | PM01 |
| Endpoint Name | Create organisation |
| Endpoint Description | Receives the request to register an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "address": "string", "bannerimage": "string", "businessname": "string", "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "manager": { "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "password": "string", "passwordexpired": true, "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string" } }</pre> |

| | |
|----------------------|---|
| | <pre> }, "postalcode": "string", "type": { "id": 0, "name": "string" }, "wallet": "string", "websiteurl": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-42: Policy Manager - get organisation

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | PM02 |
| Endpoint Name | Get an organisation |
| Endpoint Description | Receives the request to get an available organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • id: the id of the organisation |
| Request Body | None |
| Response Body | <pre> { "address": "string", "bannerimage": "string", "businessname": "string", "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "manager": { "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "password": "string", "passwordexpired": true, </pre> |

| | |
|--|---|
| | <pre> "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string", }, "postalcode": "string", "type": { "id": 0, "name": "string" }, }, "wallet": "string", "websiteurl": "string" } </pre> |
|--|---|

Table 3-43: Policy Manager - get all organisations

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | PM03 |
| Endpoint Name | Get all organisations |
| Endpoint Description | Receives the request to get all available organisations in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/ |
| HTTP method | GET |
| Request Parameters | None |
| Request Body | None |
| Response Body | <pre> [{ "address": "string", "bannerimage": "string", "businessname": "string", "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "manager": { </pre> |

| | |
|--|---|
| | <pre> "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "password": "string", "passwordexpired": true, "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string" }, "postalcode": "string", "type": { "id": 0, "name": "string" }, "wallet": "string", "websiteurl": "string" } </pre> |
|--|---|

Table 3-44: Policy Manager - update organisation

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM04 |
| Endpoint Name | Update organisation |
| Endpoint Description | Receives the request to update an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/{id} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the organisation |

| | |
|----------------------|--|
| Request Body | <pre> { "address": "string", "bannerimage": "string", "businessname": "string", "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "manager": { "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "password": "string", "passwordexpired": true, "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string" }, "postalcode": "string", "type": { "id": 0, "name": "string" }, "wallet": "string", "websiteurl": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-45: Policy Manager - suspend organisation

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | PM05 |
| Endpoint Name | Suspend an organisation |
| Endpoint Description | Receives the request to suspend an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the organisation |
| Request Body | None |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden |

Table 3-46: Policy Manager - invite users to organisation

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | PM06 |
| Endpoint Name | Invite users to an organisation |
| Endpoint Description | Receives the request to invite users to an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/{id}/users/invite |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the organisation |
| Request Body | <pre>{ "users": [{"email": "string", "firstname": "string", "lastname": "string"}] }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-47: Policy Manager - get users of organisation

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | PM06 |
| Endpoint Name | Get users of an organisation |
| Endpoint Description | Receives the request to get the users of an organisation in the ICARUS platform |

| | |
|---------------------------|---|
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/organisation/{id}/users/ |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the organisation |
| Request Body | <pre>{ "users": [{ "department": "string", "email": "string", "enabled": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "passwordexpired": true, "phone": "string", "position": "string", "username": "string" }] }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-48: Policy Manager - create user

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM07 |
| Endpoint Name | Create user |
| Endpoint Description | Receives the request to create the user of an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/user |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "organization": { "address": "string", "bannerimage": "string", "businessname": "string", </pre> |

| | |
|----------------------|---|
| | <pre> "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "postalcode": "string", "type": { "id": 0, "name": "string" }, "users": [null], "wallet": "string", "websiteurl": "string" }, "password": "string", "passwordexpired": true, "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-49: Policy Manager - update user

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | PM07 |
| Endpoint Name | Update user |
| Endpoint Description | Receives the request to update the user of an organisation in the ICARUS platform |
| Component | Policy Manager |

| | |
|---------------------------|---|
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/user/{id} |
| HTTP method | PUT |
| Request Parameters | None |
| Request Body | <pre>{ "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "passwordexpired": true, "phone": "string", "position": "string", "userRoles": ["string"], "username": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-50: Policy Manager - get user

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM08 |
| Endpoint Name | Get user |
| Endpoint Description | Receives the request to update the user of an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/user/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • Id: the id of the user |
| Request Body | None |
| Response Body | <pre>{ "department": "string", "email": "string", "enabled": true, "firstlogin": true, "firstname": "string", "id": 0, "image": "string", "lastname": "string", "organization": { "address": "string", "bannerimage": "string", </pre> |

| | |
|--|---|
| | <pre> "businessname": "string", "city": "string", "country": "string", "description": "string", "ethaddress": "string", "ethwallet": "string", "id": 0, "legalname": "string", "logoimage": "string", "postalcode": "string", "type": { "id": 0, "name": "string" }, "users": [null], "wallet": "string", "websiteurl": "string" }, "password": "string", "passwordexpired": true, "phone": "string", "position": "string", "rolesAsStringList": ["string"], "userRoles": [{ "id": 0, "role": { "id": 0, "name": "string" } }], "username": "string" } </pre> |
|--|---|

Table 3-51: Policy Manager - suspend user

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM09 |
| Endpoint Name | Suspend a user |
| Endpoint Description | Receives the request to suspend the user of an organisation in the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/user/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> Id: the id of the user |

| | |
|----------------------|---|
| Request Body | None |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden |

Table 3-52: Policy Manager - login

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | PM10 |
| Endpoint Name | Login to the platform |
| Endpoint Description | Receives the request to authorise the user of an organisation to access the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/login |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "email": "string", "password": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden |

Table 3-53: Policy Manager - logout

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | PM11 |
| Endpoint Name | Logout of the platform |
| Endpoint Description | Receives the request to logout the user of an organisation from the ICARUS platform |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/logout/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> • id: the id of the user |
| Request Body | None |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden |

| | |
|--|---|
| | <ul style="list-style-type: none"> 404 Not Found |
|--|---|

Table 3-54: Policy Manager - create access policy

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | PM12 |
| Endpoint Name | Create an access policy |
| Endpoint Description | Receives the request to create a new access policy |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/authorisation/policy |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <p><i>In the case of single rule:</i></p> <pre>{ "type": "singleRule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }</pre> <p><i>In the case of complex rules with AND or OR operand</i></p> <pre>{ "type": "complexRule", "access_rule": { "logicalOperator": "AND", "segments": [{ "type": "singlerule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }, { "type": "singlerule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }] } }</pre> |

| | |
|----------------------|--|
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |
|----------------------|--|

Table 3-55: Policy Manager - modify access policy

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | PM13 |
| Endpoint Name | Modify an access policy |
| Endpoint Description | Receives the request to modify an access policy |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/authorisation/policy/{id} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> • id: the id of the policy |
| Request Body | <p><i>In the case of single rule:</i></p> <pre>{ "type": "singleRule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }</pre> <p><i>In the case of complex rules with AND or OR operand</i></p> <pre>{ "type": "complexRule", "access_rule": { "logicalOperator": "AND", "segments": [{ "type": "singlerule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }, { "type": "singlerule", "access_rule": { "rule_name": "string", "ruleOperator": "string", "ruleOperand": "string", "value": "string" } }] } }</pre> |

| | |
|----------------------|--|
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |
|----------------------|--|

Table 3-56: Policy Manager - delete access policy

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM14 |
| Endpoint Name | Delete an access policy |
| Endpoint Description | Receives the request to delete an access policy |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/authorisation/policy/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> • id: the id of the policy |
| Request Body | None |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-57: Policy Manager - authorise access request

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | PM15 |
| Endpoint Name | Authorise access request |
| Endpoint Description | Receives the request to formulate an access control decision for an access request |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/authorisation/authorise |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "object": controlledObject, "actor": " string ", "action": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-58: Policy Manager - access control filter

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | PM16 |
| Endpoint Name | Access Control Filter |
| Endpoint Description | Receives the request to formulate an access control filter from the Query Explorer based on the authorisation of the user in order to be incorporated in the produced query towards the query engine |
| Component | Policy Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/policy-manager/filter/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the id of the user |
| Request Body | N/A |
| Response Body | An access control filter is returned in the form of “WHERE” clause else an error message is returned. |

3.12 ICARUS Storage and Indexing

3.12.1 Services Outline

The ICARUS Storage and Indexing component has a two-fold purpose: (a) to provide the effective and efficient storage solutions that enable the storage operations of the platform and to facilitate the data access operations of the rest of the components of the platform, and (b) to provide the flexible and high-performance indexing operations of the platform that support the data exploration and data query operations of the platform.

Both described functionalities are considered as core functionalities of the platform as they form part of the platform’s backbone infrastructure, on top of which all the rest of the components are operating in order to provide their functionalities. With regards to the storage capabilities of the platform, two separate storage solutions are exploited for different purposes, as presented also in the deliverable D4.1 of WP4. At first, the MongoDB storage solution is exploited in order to handle the storage of the ICARUS platform’s data assets. MongoDB is a well-established storage solution specialised in the effective storage and management of large volume of data with multiple key features such as end-to-end security, enhanced management operations and many more. The second storage solution is PostgreSQL that is exploited in order to store the user data, the data metadata and all operational data of the platform. PostgreSQL is also a well-known storage solution that is extremely flexible and robust with advanced security and sophisticated mechanism for data management. Since both storage solutions are providing core functionalities that are crucial

for the successful operation of all components, they are involved in all workflows described in deliverable D3.2 in order to support the platform's operations. Both solutions are offering an extended list of interfaces that support their core role in the platform's operations. These interfaces accessed by the Data Handler component that acts as an intermediate for the data access operations from the rest of the components of the platform.

With regards to the indexing capabilities of the platform the powerful open source enterprise search platform Solr is exploited. Solr is offering a sophisticated and efficient indexing mechanism that is utilised in order to index and support the query of information stored in the ICARUS storage offering enhanced data exploration and advanced search functionalities. Solr is offering a large list of interfaces as part of its software solution that are utilised mainly by the Query Explorer component but also by other components depending on their needs.

3.12.2 Interfaces

The interfaces of both the storage solutions and the indexing and search platform are provided by the MongoDB, PostgreSQL and Solr software solutions. Their documentation is included in their official documentation and their description is out of scope of the current deliverable.

3.13 Master Controller

3.13.1 Services Outline

The Master Controller is the component responsible for the enabling the "remote" execution of a job, as instructed by the components residing on the Core ICARUS platform, to either the On Premise Environment or the Secure and Private Space. The Master Controller is establishing a connection with the OnPremise Worker and the SecureSpace Worker in order to provide the set of instructions that should be executed locally by these workers utilising the services that are deployed locally on these tiers.

As described in deliverable D3.2, the Master Controller component is involved in the Master service that is involved in multiple workflows that require the cross-tier intercommunication for the "remote" execution of jobs. The Master Controller is tightly connected with the OnPremise Worker and the SecureSpace Worker in order to realise the Master/Worker paradigm that enables the allocation of the jobs received by the master to the respective worker based on the received instructions. As presented in the workflow describing the intercommunication of the Master Controller with the OnPremise Worker and the SecureSpace Worker in the Backend Ancillary services workflows in the deliverable D3.2, the Master Controller is offering the interface that receives the instructions for the job execution by the rest of the components of the platform. Upon receiving these instructions, the Master

Controller is routing them to the responsible worker in order to be executed locally based on the intended recipient of the instructions. The Master Controller is also responsible for monitoring the execution status of the assigned job by utilising the respective interface of the workers. Furthermore, the Master Controller provides the interface that the workers are utilising in order to report the completion of the assigned job. The Master Controller is involved in the Data Preparation workflow, as described in deliverable D3.2, in which the set instruction is compiled by each step of the workflow and are provided to Master Controller in order to be dispatched to the OnPremise Worker. Furthermore, the Master Controller is involved in the Data Analytics and Visualisations workflows during the ICARUS application execution in order to transfer the selected data assets to the Secure and Private Space but also to transfer the produced encrypted results back to the ICARUS Storage via the Data Handler. Additionally, the Master Controller is involved in the cases where data assets or the produced results of the analysis are downloaded locally or transferred in order to be decrypted for visualisation purposes.

The details of the interfaces described above are presented in the following sub-section.

3.13.2 Interfaces

In the following tables the interfaces of the Master Controller are defined.

Table 3-59: Master Controller - receive instructions

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | MC01 |
| Endpoint Name | Receive new job instructions |
| Endpoint Description | Receives the job instructions as compiled by the Core Platform components and routes these instructions to the OnPremise Worker or the SecureSpace Worker |
| Component | Master Controller |
| Endpoint URL | https://icarus_platform[:port]/api/v1/master-controller/job |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "processId": "string", "filePath": "string", "complete": true, "completed": true, "configuration": "string", "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, </pre> |

| | |
|----------------------|--|
| | <pre> "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "description": "string", "id": 0, "name": "string", "instructions": [{ } } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-60: Master Controller - receive job status

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | MC02 |
| Endpoint Name | Receive job completion status |
| Endpoint Description | Receives the job completion acknowledgement as reported by the OnPremise Worker or the SecureSpace Worker |
| Component | Master Controller |
| Endpoint URL | https://icarus_platform[:port]/api/v1/master-controller/job/{id}/status |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> • id: the job identifier |
| Request Body | <pre> { "processId": "string", "status": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-61: Master Controller - transfer data

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | MC03 |
| Endpoint Name | Transfer data |
| Endpoint Description | Handles the transferring request of the specified dataset from the core ICARUS storage to a Secure and Private space as provided by the Data Handler. The Master Controller informs the SecureSpace Worker for the dataset transfer, once the file transfer is ready to be executed by the appropriate backend process executed within the context of Data Handler. |
| Component | Master Controller |

| | |
|---------------------------|--|
| Endpoint URL | https://icarus_platform[:port]/api/v1/master-controller/transfer-data/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-62: Master Controller - upload results

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | MC04 |
| Endpoint Name | Upload results from Secure and Private Space |
| Endpoint Description | Handles the upload of the data analysis results in the ICARUS platform |
| Component | Master Controller |
| Endpoint URL | https://icarus_platform[:port]/api/v1/master-controller/upload-data/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | Header: Content-Type: multipart/form-data; <pre>{ "dataname": "string", "id": "string", "user": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 201 Created 401 Unauthorised 403 Forbidden 404 Not Found |

3.14 OnPremise Worker and SecureSpace Worker

3.14.1 Services Outline

The OnPremise Worker and the SecureSpace Worker are the components that are undertaking the local execution of the jobs as instructed by the Master Controller, utilising the deployed services on their local running environment.

As described in deliverable D3.2, the OnPremise Worker and the SecureSpace Worker are involved in the Worker service that is a complementary service of the Master service and is involved in multiple workflows that realise the “remote” job execution. As explained also in section 3.13, both workers are tightly connected with the Master Controller and following the

Master/Worker paradigm, the workers receive the instructions from the Master Controller and execute the assigned job. Following the workflow that was presented in the Ancillary Backend services workflows, the workers provide the interface in order to receive the set of instructions from the Master Controller and interpret these instructions in order to request the execution of the local services via their respective interfaces. Additionally, the workers are monitoring the execution status of the assigned tasks from the job via the dedicated interfaces of the selected services and provide an interface for the Master Controller to obtain this status. Finally, the workers provide an interface that the local services are utilising in order to report the completion of their task. Once all tasks of the job are completed, the workers utilise the interface provided by the Master Controller to report the job completion. As with the Master Controller, the OnPremise Worker is involved in the Data Preparation workflow and receives the compiled set of instructions that is interpreted and executed by the respective local services. Furthermore, the SecureSpace Worker is involved in the Data Analytics and Visualisations workflows for the transfer of the utilised data assets during the ICARUS application execution and in the transferring of the produced encrypted results once the data analysis is completed. Finally, both workers are involved in the cases where data assets are downloaded locally and the SecureSpace Worker is additionally involved in the transferring of the data asset during the visualisation process.

The details of the interfaces described above are presented in the following sub-section.

3.14.2 Interfaces

In the following tables the interfaces of the On Premise Worker and SecureSpace Worker are defined.

Table 3-63: On Premise Worker and SecureSpace Worker - receive instructions

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | WO01 |
| Endpoint Name | Receive new job instructions by the Master Controller |
| Endpoint Description | Receives the job instructions from the Master Controller and distribute the execution to the local services |
| Component | OnPremise Worker, SecureSpace Worker |
| Endpoint URL | https://hostname[:port]/v1/workers/job |
| HTTP method | POST |
| Request Parameters | None |

| | |
|----------------------|---|
| Request Body | <pre>{ "processId": "string", "filePath": "string", "complete": true, "completed": true, "configuration": "string", "created": { "date": 0, "day": 0, "hours": 0, "minutes": 0, "month": 0, "nanos": 0, "seconds": 0, "time": 0, "timezoneOffset": 0, "year": 0 }, "description": "string", "id": 0, "name": "string", "instructions": [{ }] }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-64: On Premise Worker and SecureSpace Worker - retrieve job status

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | WO02 |
| Endpoint Name | Retrieve job status |
| Endpoint Description | Receives the job status per task as reported by the corresponding services that have undertaken the tasks for a job |
| Component | OnPremise Worker, SecureSpace Worker |
| Endpoint URL | https://hostname[:port]/v1/workers/job/{id}/status |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • id: the job identifier |
| Request Body | <pre>{ "processId": "string", "status": { "mapping" : "string", "cleaning": "string", "encryption": "string", "anonymisation": "string" } }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 400 Bad Request • 404 Not Found |

Table 3-65: On Premise Worker and SecureSpace Worker - receive task status

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | WO03 |
| Endpoint Name | Receive task completion status |
| Endpoint Description | Receives the task completion acknowledgement as reported by the corresponding service that has undertaken the task for a job |
| Component | OnPremise Worker, SecureSpace Worker |
| Endpoint URL | https://hostname[:port]/v1/workers/job/{id}/task/{task_id}/status |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the job identifier task_id: the task identifier |
| Request Body | <pre>{ "processId": "string", "taskId": "string", "status": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> 200 OK 400 Bad Request 404 Not Found |

Table 3-66: On Premise Worker and SecureSpace Worker - upload results

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | WO04 |
| Endpoint Name | Upload results from Secure and Private Space |
| Endpoint Description | Handles the upload of the results of the data analysis in the ICARUS platform. Internally, the Master Controller is invoked in order to setup the internal procedure for the data transfer of the results and their storage in the ICARUS platform. |
| Component | SecureSpace Worker |
| Endpoint URL | https://hostname[:port]/v1/workers/upload-data/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> id: the dataset id |
| Request Body | <p>Header: Content-Type: multipart/form-data;</p> <pre>{ "dataname": "string", "id": "string", "user": "string" }</pre> |

| | |
|----------------------|---|
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |
|----------------------|---|

Table 3-67: On Premise Worker and SecureSpace Worker – receive data

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | WO05 |
| Endpoint Name | Receive data |
| Endpoint Description | Handles the transferring request of the specified dataset from the core ICARUS storage to a Secure and Private space as provided by the Master Controller. The SecureSpace Worker is informed by the Master Controller for the dataset transfer and the appropriate backend process is executed for dataset transfer. |
| Component | SecureSpace Worker |
| Endpoint URL | https://hostname[:port]/v1/workers/transfer-data/{id} |
| HTTP method | POST |
| Request Parameters | <ul style="list-style-type: none"> • id: the dataset id |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

3.15 Query Explorer

3.15.1 Services Outline

As described in deliverables D3.1 and D3.2, the Query Explorer is the component that offers advanced dataset exploration and discoverability functionalities to the users of the ICARUS platform. It should be noted that the processes offered by the component extend beyond typical search functionalities found in other data marketplaces and relevant platforms, due to the complexities introduced by the fact that datasets are uploaded partially encrypted in ICARUS. The Query Explorer is the main acting component in the Data Exploration workflows presented in deliverable D3.2, which commonly also act as an entry point to the Asset Brokerage workflows.

The component enables users to define search criteria in a flexible manner and thus facilitates the discoverability of potentially interesting datasets, dataset subsets and dataset combinations. Specifically, it allows a user to:

- select the fields of the common ICARUS data model that should be present in the datasets that will be included in the results,
- express and apply filters based on the metadata of the datasets that will be included in the results,
- express and apply filters on the actual data of the datasets that will be included in the results. These filters are only available for unencrypted data columns in accordance also to the instructions of the dataset provider regarding including or excluding them from the dataset index.

The Query Explorer transforms the defined user selections and filters into a Solr query, which is seamlessly combined with an appropriate access policy filter that is applied on the Solr query (with the help of the Policy Manager), to ensure that the search will be performed only on datasets that the current user in the current context is eligible to receive as results (candidates for acquisition). Finally, the component is also responsible for showing the results to the user in an intuitive and comprehensive manner.

The details of the interfaces described above are presented in the following sub-section.

3.15.2 Interfaces

Query Explorer leverages the Solr API, which is out of scope to be documented here. Therefore the interfaces are documented below in a higher level of abstraction. It should be noted however that the invocation of the query creation service results in the background in the invocation of other services that ultimately result in the creation and execution of a Solr query. One of these services invoked in the background is Table 3-58 documented in Section 3.11.2.

Table 3-68: Query Explorer - create and execute query

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | QE01 |
| Endpoint Name | Create and execute a query |
| Endpoint Description | Create a new query based on selected fields and defined filters, execute it and return the results |
| Component | Query Explorer |
| Endpoint URL | https://icarus_platform[:port]/api/v1/query/ |
| HTTP method | POST |
| Request Parameters | N/A |

| | |
|----------------------|--|
| Request Body | <i>A set of fields and filters as set per the requestor:</i> <pre>{ "fields": [{ }], "filters": [{ }] }</pre> |
| Response Body | <i>A set of results from the query execution on the respective Solr API:</i> <pre>{ "results":[] }</pre> |

Table 3-69: Query Explorer - get query definition

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | QE02 |
| Endpoint Name | Get query definition |
| Endpoint Description | Get the definition of a specific query (i.e. its fields and filters) |
| Component | Query Explorer |
| Endpoint URL | https://icarus_platform[:port]/api/v1/query/{id}/ |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> Id: the id of the query |
| Request Body | N/A |
| Response Body | <i>A set of fields and filters as set in the query creation:</i> <pre>{ "fields": [{ }], "filters": [{ }] }</pre> |

Table 3-70: Query Explorer - get updated query results

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | QE03 |
| Endpoint Name | Get updated query results |
| Endpoint Description | Re-execute a specific query and return the updated results |
| Component | Query Explorer |
| Endpoint URL | https://icarus_platform[:port]/api/v1/query/{id}/results |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> Id: the id of the query |
| Request Body | N/A |

| | |
|----------------------|---|
| Response Body | <i>A set of results from the query execution on the respective Solr API:</i> <pre>{ "results":[] }</pre> |
|----------------------|---|

Table 3-71: Query Explorer - delete query

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | QE04 |
| Endpoint Name | Delete query |
| Endpoint Description | Delete a specific query |
| Component | Query Explorer |
| Endpoint URL | https://icarus_platform[:port]/api/v1/query/{id}/ |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> Id: the id of the query |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 400 Bad Request 404 Not Found |

Table 3-72: Query Explorer - get query history

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | QE05 |
| Endpoint Name | Get query history for user |
| Endpoint Description | Get all saved queries of the user |
| Component | Query Explorer |
| Endpoint URL | https://icarus_platform[:port]/api/v1/query/user/{id}/all |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> id: the user id |
| Request Body | N/A |
| Response Body | <pre>{ "queries": [] }</pre> |

3.16 Recommender

3.16.1 Services Outline

The Recommender is the component responsible for providing accurate suggestions of data assets that exist in the ICARUS repository to the stakeholders, based on their preferences. As described in deliverable D3.2, the Recommender is involved in the Data Recommendation service that involves two distinct phases: the offline training and the execution phases. In each phase, the Recommender interacts with various other components via REST APIs, provided by the related components.

During the offline training phase, the Recommender interacts with the Data Handler, that is offering a layer above the ICARUS storage, in order to collect relevant information from the ICARUS Storage about the data assets and the users' behaviour that is stored there. Accessing such information enables the Recommender to accomplish one of its main functionalities, which is to improve its recommendations by re-training its algorithm periodically. More precisely, the Recommender collects from the ICARUS Storage, again via the Data Handler, the users' IDs and the users' history (i.e. searches, views, favourites, purchases) in order to train its model for providing user-based recommendations of data assets to the users. In addition to this, the Recommender collects the data assets' IDs and their categories (e.g. weather, flight delays, etc.) in order to be trained for providing a set of recommended data assets using an item-based approach.

The execution phase aims at providing recommendations of data assets to a specific user and it is triggered when a stakeholder uses the ICARUS Data Exploration service which is handled by the Query Explorer component, so as to search for data assets. The Recommender's REST API is called from the Query Explorer, receiving as parameters the user's ID and the data assets' IDs that the user is currently viewing. In addition, the Recommender interacts with the ICARUS Storage, via the Data Handler, in order to collect the user's preferences which are manually set by each user in their profiles, as well as semantic metadata of the data assets related to their entities which are derived from the ICARUS Ontology. Finally, after collecting and processing the previously mentioned information, the Recommender responds to the Query Explorer with a list of recommended data assets' IDs.

As described above, the Recommender is exploiting the interfaces of the Data Handler in order to retrieve the required information for its operations. The Recommender exposes one interface to the Query Explorer in order to generate the user-specific recommendations that will be offered. The details of this interface is presented in the following sub-section.

3.16.2 Interfaces

In the following table the interface of the Recommender is defined.

Table 3-73: Recommender - data assets recommendations

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | REC01 |
| Endpoint Name | Data assets recommendations. |
| Endpoint Description | Generates recommendations of data assets for a specific user. |
| Component | Recommender |
| Endpoint URL | https://icarus_platform[:port]/api/v1/recommender/ |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <p>The request body is in JSON format and contains the user ID of the user (type: "string") that will receive the recommendations and a list data assets IDs (the list can be empty or can have multiple "strings" that represent data assets IDs). For example:</p> <pre>{ "user_id": "U12345678", "datasets_id": ["DS67438434", "DS05855493", ...] }</pre> |
| Response Body | <p>The response body is in JSON format and contains a list of the recommended data assets IDs (each ID is of type "string"). For example:</p> <pre>{ "recommended_datasets": ["DS95855433", "DS99882437", "D550087652", ...] }</pre> |

3.17 Analytics and Visualisation Workbench

3.17.1 Services Outline

The Analytics and Visualisation Workbench is the component that enables the design, execution and monitoring of the data analytics workflows within the ICARUS platform. To meet its goal, the Analytics and Visualisation Workbench is handling several aspects that span from the design of an ICARUS application, to the execution of this application and the visualisation of the produced results in a meaningful way.

As described in deliverables D3.1 and D3.2, the Analytics and Visualisation Workbench is involved in the following main functionalities of the ICARUS platform: (a) the creation of a new ICARUS application which a set of defined datasets the user owns or has legitimate access, data analysis algorithms with their corresponding parameters, and a set of selected

visualisations, (b) the immediate and/or the scheduled execution of an ICARUS application, (c) the generation of visualisations of the produced results from the execution of the ICARUS application and (d) the export of the produced results. Therefore, the Analytics and Visualisation Workbench is the key part of the Data Analytics and Visualisations workflow and provides all the functionalities that compose the Data Analytics Service as presented in deliverable D3.2.

Behind the scenes, the Analytics and Visualisation Workbench consists of three fundamental sub-components that are integrated in order to provide the aforementioned functionalities: (a) the simple and intuitive user interface that is offered to the users of the ICARUS platform, (b) the repository for the implemented algorithms and (c) a microservice that exposes a RESTful API that enables the mediation between the frontend and the various backend components that the Analytics and Visualisation Workbench interacts with such as the Job Scheduler and Execution Engine, the Resource Orchestrator, the Data Handler and the BDA Application Catalogue.

During the creation of the ICARUS application, the Analytics and Visualisation Workbench interacts with the BDA Application Catalogue, as depicted in the ICARUS application design workflow which is documented in deliverable D3.2, in order to retrieve the existing ICARUS applications or store the newly designed ICARUS application and its metadata. The Analytics and Visualisation Workbench exploits the interfaces provided by the BDA Application Catalogue in order to perform the whole lifecycle management of the ICARUS applications.

During the ICARUS application execution, the Analytics and Visualisation Workbench is interacting with the Resource Orchestrator and the Job Scheduler backend components, as depicted in the ICARUS application execution workflow which is documented also in deliverable D3.2. Specifically, the Analytics and Visualisation Workbench firstly ensures the existence of the Secure and Private Space by interacting with the Resource Orchestrator and its provided interfaces. Afterwards, the Analytics and Visualisation Workbench interacts with the Job Scheduler in order to either initiate the ICARUS application execution or to schedule its execution via the respective interfaces offered by the Job Scheduler. Once the results of the execution are available, the Analytics and Visualisation Workbench is informed and they are made available to the user for visualisation purposes.

The Analytics and Visualisation Workbench is offering the generation of a variety of visualisations on top of the produced results or the option to export and download the results locally. During this process, the Analytics and Visualisation Workbench is interacting with the Data Handler and the BDA Application Catalogue, as depicted in the results visualisation workflow that is documented also in deliverable D3.2. In case of the visualisation generation,

the Analytics and Visualisation Workbench after receiving the visualisation configuration from the user, it requests from the Data Handler the required unencrypted results via the provided interface. The Data Handler undertakes the orchestration of the process, as it interacts with the Policy Manager to validate the access request, as well as with the Master Controller in order to initiate the decryption process that is executed within the Secure and Private Space. Finally, it provides the results to the Analytics and Visualisation Workbench that generates the requested visualisation and updates the application's visualisation parameters in the BDA Application Catalogue. In case of the export or downloading of the produced results, the Analytics and Visualisation Workbench requests from the Data Handler the preparation of the results and the download process is executed followed by the respective decryption process. As described above, the Analytics and Visualisation Workbench is mainly providing the user interface that is orchestrating the rest of the components in order to provide the described functionalities. Thus, the Analytics and Visualisation Workbench is mainly exploiting the interfaces of the rest of the components. However, internally a set of interfaces is provided by the component in order to facilitate its internal operations. These operations include the lifecycle management of the implemented data analytics algorithms and the ICARUS applications with the help of the BDA Applications Catalogue, as well as the interactions with the Job Scheduler for the execution or scheduled execution of the ICARUS applications. The details of the internal interfaces offered by the Analytics and Visualisation Workbench, as described above, are presented in the following sub-section.

3.17.2 Interfaces

In the following tables the interfaces of the Analytics and Visualisation Workbench are defined.

Table 3-74: Analytics and Visualisation Workbench – register algorithm

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | AVW01 |
| Endpoint Name | Registration Controller / Algorithms |
| Endpoint Description | It allows to register algorithms in the repository and to obtain a list of the registered algorithms. The algorithms will be used to build up the ICARUS Applications. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/algorithms |
| HTTP method | GET, POST |
| Request Parameters | None |

| | |
|----------------------|--|
| Request Body | <pre>{ "catalogues": list, "area": area, "name": "string", "metrics": set, "mode": mode, "version": "string", "description": "string", "properties": "string", "url": "string" }</pre> |
| Response Body | <p>In case of POST:</p> <pre>{ "algorithm": { "id": long, "catalogues": list, "area": area, "name": "string", "metrics": set, "mode": mode, "version": "string", "description": "string", "properties": set, "url": "string" } }</pre> <p>In case of GET returns a list of Algorithms.</p> |

Table 3-75: Analytics and Visualisation Workbench – get, delete algorithm

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | AVW02 |
| Endpoint Name | Registration Controller / Algorithms |
| Endpoint Description | It allows to obtain or delete a registered algorithm from the repository. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/algorithms/{id} |
| HTTP method | DELETE, GET |
| Request Parameters | <ul style="list-style-type: none"> Id: the unique identifier of the registered algorithm |
| Request Body | None |
| Response Body | <p>In case of GET:</p> <pre>{ "algorithm": { "id": long, "catalogues": list, "area": area, "name": "string", "metrics": set, </pre> |

| | |
|--|---|
| | <pre> “mode”: mode, “version”: “string”, “description”: “string”, “properties”: set, “url”: “string” } } None in case of DELETE. </pre> |
|--|---|

Table 3-76: Analytics and Visualisation Workbench – register application

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | AVW03 |
| Endpoint Name | Registration of Applications |
| Endpoint Description | It allows to register BDA Applications interacting with the BDA Application Catalogue |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/applications |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre> { “name”: “string”, “description”: “string”, “created”: date, “updated”: date, “metadata”: { “id”: 1, “categories”: [“string”, “string”], “tags”: “string”, “accessibility”: “string”, “version”: “string”, “license”: “string”, “privacylevel”: “string”, “allowedmodify”: true, “allowedexcept”: true, “allowedannotate”: true, “allowedaggregate”: true, “attribution”: “string”, “reproduction”: “string”, “distribution”: “string”, “targetpurpose”: “string”, “targetindustry”: “string”, “organizationcategoryrecontext”: “string”, “calculationscheme”: “string”, </pre> |

| | |
|----------------------|---|
| | <pre> "amount": "string", "paymentmethod": "string", "policy": "string", ... }, "dataset": { "id":1, "type":"input", "name":"string" ... }, "workflow": { "id": "string", "name"string", "definition": "string", "algorithms":[{ "id": "string", "category": "string", "name": "string", "version": "string", "description": "string", "framework": { "id": "string", "name": "string", "version": "string" }, "properties": [...], "url": "string" }, { ...}] }, "user": user } </pre> |
| Response Body | 200 OK and the same information as with the request body for confirmation |

Table 3-77: Analytics and Visualisation Workbench – get applications

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | AVW04 |
| Endpoint Name | Retrieve Applications |
| Endpoint Description | It allows to obtain a list of the registered BDA applications. |

| | |
|---------------------------|--|
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/applications |
| HTTP method | GET |
| Request Parameters | None |
| Request Body | N/A |
| Response Body | <pre>[{ "name": "string", "description": "string", "created": date, "updated": date, "metadata": { "id": 1 "categories": ["string", "string"], "tags": "string", "accessibility": "string", "version": "string", "license": "string", "privacylevel": "string", "allowedmodify": true, "allowedexcept": true, "allowedannotate": true, "allowedaggregate": true, "attribution": "string", "reproduction": "string", "distribution": "string", "targetpurpose": "string", "targetindustry": "string", "organizationcategoryrecontext": "string", "calculationscheme": "string", "amount": "string", "paymentmethod": "string", "policy": "policy", ... }, "dataset": { "id":1, "type":"input", "name":"string" ... }, "workflow": { "id": "string", "name": "string", </pre> |

| | |
|--|--|
| | <pre> "definition": "string", "algorithms": [{ "id": "string", "category": "string", "name": "string", "version": "string", "description": "string", "framework": { "id": "string", "name": "string", "version": "string" }, "properties": [...], "url": "string" }, { ...}] }, "user": "string" }] </pre> |
|--|--|

Table 3-78: Analytics and Visualisation Workbench – get application

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | AVW05 |
| Endpoint Name | Get Application |
| Endpoint Description | It allows to obtain a registered BDA application. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/applications/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> Id: the unique identifier of the registered application |
| Request Body | N/A |
| Response Body | <pre> { "application": { { "name": "string", "description": "string", "created": date, "updated": date, "metadata": { "id": 1 } } } } </pre> |

```

"categories": ["string", "string"],
"tags": "string",
"accessibility": "string",
"version": "string",
"license": "string",
"privacylevel": "string",
"allowedmodify": true,
"allowedexcept": true,
"allowedannotate": true,
"allowedaggregate": true,
"attribution": "string",
"reproduction": "string",
"distribution": "string",
"targetpurpose": "string",
"targetindustry": "string",
"organizationcategoryrecontext": "string",
"calculationscheme": "string",
"amount": "string",
"paymentmethod": "string",
"policy": "string",
...
},
"dataset": {
  "id": 1,
  "type": "input",
  "name": "string",
  ...
},
"workflow": {
  "id": "string",
  "name": "string",
  "definition": "string",
  "algorithms": [
    {
      "id": "string",
      "category": "string",
      "name": "string",
      "version": "string",
      "description": "string",
      "framework": {
        "id": "string",
        "name": "string",
        "version": "string"
      },
    },
    "properties": [".", "..."],

```

| | |
|--|---|
| | <pre> "url": "string" }, { ...}] }, "user": user }</pre> |
|--|---|

Table 3-79: Analytics and Visualisation Workbench – delete application

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AVW06 |
| Endpoint Name | Delete an Application |
| Endpoint Description | It allows to delete a registered BDA application. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/registration/applications/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> Id: the unique identifier of the registered application |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 400 Bad Request 404 Not Found |

Table 3-80: Analytics and Visualisation Workbench – schedule job

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AVW07 |
| Endpoint Name | Scheduling Controller Specific Job |
| Endpoint Description | It allows to interact with the Job Scheduler and Execution engine deployed within the Secure and Private Space. The services behind this API adds business logic to retrieve the proper destination of the request. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/scheduling/entries/{id} |
| HTTP method | GET, PUT, DELETE |
| Request Parameters | <ul style="list-style-type: none"> Id: the unique identifier of the registered job |
| Request Body | <pre> { "name": string, "application_id": long, "user_id": integer, "timezone": "string", "recurrent": boolean,</pre> |

| | |
|----------------------|---|
| | <pre> “cron_exp”: “string”, “start_time”: datetime, “end_time”: datetime </pre> |
| Response Body | <pre> { “name”: “string”, “application_id”: LONG, “user_id”: INTEGER, “timezone”: “string”, “recurrent”: BOOLEAN, “cron_exp”: “string”, “start_time”: DATETIME, “end_time”: DATETIME } </pre> |

Table 3-81: Analytics and Visualisation Workbench – add job

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | AVW08 |
| Endpoint Name | Scheduling Controller Add job |
| Endpoint Description | It allows to interact with the Job Scheduler and Execution engine deployed within the secure and private space. The services behind this API adds business logic to retrieve the proper destination of the request. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/scheduling/entries |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre> { “name”: “string”, “application_id”: long, “user_id”: long, “timezone”: “string”, “recurrent”: boolean, “cron_exp”: “string”, “start_time”: datetime, “end_time”: datetime } </pre> |
| Response Body | <pre> { “name”: “string”, “application_id”: long, “user_id”: long, “timezone”: “string”, “recurrent”: boolean, “cron_exp”: “string”, “start_time”: datetime, “end_time”: datetime } </pre> |

Table 3-82: Analytics and Visualisation Workbench – get all jobs

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AVW09 |
| Endpoint Name | Scheduling Controller – Get Jobs |
| Endpoint Description | It allows to interact with the Job Scheduler and Execution engine deployed within the secure and private space. The services behind this API adds business logic to retrieve the proper destination of the request. |
| Component | Analytics and Visualization Workbench |
| Endpoint URL | https://icarus_platform[:port]/api/v1/analytics/scheduling/entries |
| HTTP method | GET |
| Request Parameters | N/A |
| Request Body | N/A |
| Response Body | <pre>[{ "id": long, "name": "string", "application_id": long, "user_id": long, "recurrent": boolean, "cron_exp": "string", "created_time": datetime, "updated_time": datetime }, ...]</pre> |

3.18 BDA Application Catalogue

3.18.1 Services Outline

The BDA Application Catalogue is a complementary component providing the repository where all ICARUS applications are stored, accessed and maintained. The BDA Application Catalogue enables the reuse, update and sharing of these applications (under a license defined by the owner of the application) facilitating the functionalities offered to the users of the platform through the user interface of the Analytics and Visualisation Workbench.

As described before, an ICARUS application is designed within the context of Data Analytics service via the user interface offered by the Analytics and Visualisation Workbench and constitutes a set metadata for the selected datasets, the selected data analytics algorithms and the selected visualisation types and their parameters. The BDA Application Catalogue provides a group of CRUD RESTful-API interfaces for the purpose of managing these ICARUS Applications. In accordance with the workflows described in the deliverable D3.2, the BDA Application Catalogue is involved in the ICARUS application design workflow providing the

repository where the designed application is stored and in the results visualisation workflow in order to store the updates on the application. In general, the BDA Application Catalogue is interacting only with the Analytics and Visualisation Workbench and is involved only in the Data Analytics service.

The details of the interfaces offered by the BDA Application Catalogue, as described above, are presented in the following sub-section.

3.18.2 Interfaces

In the following tables the interfaces of the Analytics and Visualisation Workbench are defined.

Table 3-83: BDA Application Catalogue - application creation

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AC01 |
| Endpoint Name | Application Creation |
| Endpoint Description | The service allows to add a new ICARUS Application |
| Component | BDA Application Catalogue |
| Endpoint URL | https://icarus_platform[:port]/api/v1/app-catalogue/applications |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "name": "string", "description": "string", "created": date, "updated": date, "metadata": { "id": 1 "categories": ["string", "string"], "tags": "string", "accessibility": "string", "version": "string", "license": "string", "privacylevel": "string", "allowedmodify": true, "allowedexcept": true, "allowedannotate": true, "allowedaggregate": true, "attribution": "string", "reproduction": "string", "distribution": "string", "targetpurpose": "string",</pre> |

| | |
|----------------------|--|
| | <pre> "targetindustry": "string", "organizationcategoryrecontext": "string", "calculationscheme": "string", "amount": "string", "paymentmethod": "string", "policy": "string", ... }, "dataset": { "id": 1, "type": "input", "name": "string" ... }, "workflow": { "id": "string", "name": "string", "definition": "string", "algorithms": [{ "id": "string", "category": "string", "name": "string", "version": "string", "description": "string", "framework": { "id": "string", "name": "string", "version": "string" }, "properties": [...], "url": "string" }, { ...}] }, "user": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 201 Created • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-84: BDA Application Catalogue - get application

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AC02 |
| Endpoint Name | Applications |
| Endpoint Description | The service allows to get a specific ICARUS Application |
| Component | BDA Application Catalogue |
| Endpoint URL | https://icarus_platform[:port]/api/v1/app-catalogue/applications/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> id : The id of the application that has to be get |
| Request Body | N/A |
| Response Body | <pre>{ "name": "string", "description": "string", "created": date, "updated": date, "metadata": { "id": 1 "categories": ["string", "string"], "tags": "string", "accessibility": "string", "version": "string", "license": "string", "privacylevel": "string", "allowedmodify": true, "allowedexcept": true, "allowedannotate": true, "allowedaggregate": true, "attribution": "string", "reproduction": "string", "distribution": "string", "targetpurpose": "string", "targetindustry": "string", "organizationcategoryrecontext": "string", "calculationscheme": "string", "amount": "string", "paymentmethod": "string", "policy": "string", ... }, "dataset": { "id": 1, "type": "input", </pre> |

| | |
|--|--|
| | <pre> "name": "string", ... }, "workflow": { "id": "string", "name": "string", "definition": "string", "algorithms": [{ "id": "string", "category": "string", "name": "string", "version": "string", "description": "string", "framework": { "id": "string", "name": "string", "version": "string" }, "properties": [...], "url": "string" }, { ...}] }, "user": "string" } </pre> |
|--|--|

Table 3-85: BDA Application Catalogue - update application

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AC03 |
| Endpoint Name | Applications |
| Endpoint Description | The service allows to update a specific ICARUS Application |
| Component | BDA Application Catalogue |
| Endpoint URL | https://icarus_platform[:port]/api/v1/app-catalogue/applications/{id} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: The id of the application that has to be update |
| Request Body | <pre> { "name": "My BDA Application edited", "description": "Free Text", "created": date, "updated": date, </pre> |

```

“metadata”: {
  “id”: 1,
  “categories”: [“string”, “string”],
  “tags”: “string”,
  “accessibility”: “string”,
  “version”: “string”,
  “license”: “string”,
  “privacylevel”: “string”,
  “allowedModify”: True,
  “allowedExcept”: True,
  “allowedAnnotate”: True,
  “allowedAggregate”: True,
  “attribution”: “Allowed”,
  “reproduction”: “Allowed”,
  “distribution”: “Allowed”,
  “targetPurpose”: “Academic/Scientific”,
  “targetIndustry”: “string”,
  “organizationCategoryReContext”: “Allowed”,
  “calculationScheme”: “string”,
  “amount”: “100,00”,
  “paymentMethod”: “BankTransfer”,
  “policy”: “string”,
  ...
},
“dataSet”: {
  “id”:1,
  “type”: “INPUT”,
  “name”: “avio.csv”,
  ...
},
“workflow”: {
  “id”:1,
  “name”: “My BDA Application Workflow”,
  “definition”: “spark-kmeans-model | spark-predictive-analytics”,
  “algorithms”: [
    {
      “id”: 51,
      “category”: “MACHINE-LEARNING”,
      “name”: “spark-kmeans-model”,
      “version”: “1.0.1”,
      “description”: “Task to train a KMeans model with Spark ML”,
      “framework”: {
        “id”: 5,
        “name”: “Spark”,
        “version”: “2.4.0”
      }
    }
  ]
}

```

| | |
|----------------------|--|
| | <pre> }, "properties": [...], "url": "docker://nodo1.toreador.org:8082/spark-kmeans-model:1.0.0" }, { "id": 53, "category": "MACHINE-LEARNING", "name": "spark-predictive-analytics", "version": "1.0.0", "description": "Task to make prediction using a trained model", "framework": { "id": 5, "name": "Spark", "version": "2.4.0" }, "properties": [...], "url": "docker://nodo1.toreador.org:8082/spark-predictive- analytics:1.0.0" }] }, "user": "string" } </pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-86: BDA Application Catalogue - delete application

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | AC04 |
| Endpoint Name | Applications |
| Endpoint Description | The service allows to delete a specific ICARUS Application |
| Component | BDA Application Catalogue |
| Endpoint URL | https://icarus_platform[:port]/api/v1/app-catalogue/applications/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> • id: The id of the application that has to be deleted |
| Request Body | N/A |
| Response Body | N/A |

Table 3-87: BDA Application Catalogue - get all applications

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | AC05 |
| Endpoint Name | Applications |
| Endpoint Description | The service allows to list the ICARUS Applications. |
| Component | BDA Application Catalogue |
| Endpoint URL | https://icarus_platform[:port]/api/v1/app-catalogue/applications |
| HTTP method | GET |
| Request Parameters | N/A |
| Request Body | N/A |
| Response Body | <pre>[{ "name": "string", "description": "string", "created": date, "updated": date, "metadata": { "id": 1, "categories": ["string", "string"], "tags": "string", "accessibility": "string", "version": "string", "license": "string", "privacylevel": "string", "allowedmodify": true, "allowedexcept": true, "allowedannotate": true, "allowedaggregate": true, "attribution": "string", "reproduction": "string", "distribution": "string", "targetpurpose": "string", "targetindustry": "string", "organizationcategoryrecontext": "string", "calculationscheme": "string", "amount": "string", "paymentmethod": "string", "policy": "string", ... }, "dataset": { "id": 1,</pre> |

| | |
|--|---|
| | <pre> "type": "input", "name": "string", ... }, "workflow": { "id": "string", "name": "string", "definition": "string", "algorithms": [{ "id": "string", "category": "string", "name": "string", "version": "string", "description": "string", "framework": { "id": "string", "name": "string", "version": "string" }, "properties": [...], "url": "string" }, { ...}] }, "user": "string" }}</pre> |
|--|---|

3.19 Resource Orchestrator

3.19.1 Services Outline

The Resource Orchestrator is the component responsible for the deployment of the Secure and Private Space that is providing the “sandboxed” environment where the data analysis is performed in a secure and isolated manner. To this end, the Resource Orchestrator is responsible for the provisioning and management of the Secure and Private Spaces by connecting to the underlying infrastructure in order to perform monitoring and management of the available resources, to provision the dedicated virtual machines, as well as to deploy and monitor the required services of the Secure and Private Spaces.

As described in deliverable D3.2, the Resource Orchestrator is involved in the Resource Orchestration service that is included in the Secure and Private Space provisioning / stoppage workflow included in the Backend Ancillary services workflows. In particular, the Resource Orchestrator is interacting with the Analytics and Visualisation Workbench in order to

provision the set of virtual machines that are required for the Secure and Private Space deployment. For this purpose, the Resource Orchestrator is offering the interface that receives the deployment request of the virtual machines from the Analytics and Visualisation Workbench. Once the virtual machines are provisioned, the Resource Orchestrator receives a request via the provided interface for the deployment of the services that are operating in the Secure and Private Space. Once the deployment of the services is completed, the Analytics and Visualisation Workbench is informed by the Resource Orchestrator in order to start the data analysis. Finally, the Resource Orchestrator is providing the interface that receives the request for the stoppage or shutdown of the Secure and Private Space. The described process is executed as part of the ICARUS application execution workflow included in the Data Analytics and Visualisation workflows.

The details of the interfaces described above are presented in the following sub-section.

3.19.2 Interfaces

In the following tables the interfaces of the Resource Orchestration are defined

Table 3-88: Resource Orchestrator - deploy secure and private space

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | RO01 |
| Endpoint Name | Deploy the Secure and Private Space |
| Endpoint Description | Receives the request to deploy the Secure and Private Space for the specific user of an organisation |
| Component | Resource Orchestrator |
| Endpoint URL | https://icarus_platform[:port]/api/v1/resource-orchestrator/deploy/ |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "username": "string", "organization_id": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-89: Resource Orchestrator - stop secure and private space

| ICARUS Technical Interface |
|----------------------------|
|----------------------------|

| | |
|-------------------------------|--|
| Technical Interface ID | RO02 |
| Endpoint Name | Stop the Secure and Private Space |
| Endpoint Description | Receives the request to stop the Secure and Private Space for the specific user of an organisation |
| Component | Resource Orchestrator |
| Endpoint URL | https://icarus_platform[:port]/api/v1/resource-orchestrator/deploy/stop |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "username": "string", "organization_id": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

Table 3-90: Resource Orchestrator - deploy services on secure and private space

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | RO03 |
| Endpoint Name | Deploy the services for Secure and Private Space |
| Endpoint Description | Receives the request to deploy the services for the Secure and Private Space |
| Component | Resource Orchestrator |
| Endpoint URL | https://icarus_platform[:port]/api/v1/resource-orchestrator/deploy/services |
| HTTP method | POST |
| Request Parameters | None |
| Request Body | <pre>{ "username": "string", "organization_id": "string" }</pre> |
| Response Body | <ul style="list-style-type: none"> • 200 OK • 401 Unauthorised • 403 Forbidden • 404 Not Found |

3.20 Jobs Scheduler and Execution Engine

3.20.1 Services Outline

The Jobs Scheduler and Execution Engine is the component that enables the execution of the ICARUS applications that were designed in the Analytics and Visualisation Workbench within the context of the Secure and Private Space of the ICARUS platform.

As described in deliverables D3.1 and D3.2, the Jobs Scheduler and Execution Engine component performs the background operations for the allocation and management of the jobs that need to be executed in the context of an ICARUS application execution that includes: (a) the deployment and management of the Execution Cluster with the help of the Resource Orchestrator, (b) the execution (immediate or scheduled) of the data analysis on the Execution Engine, and (c) the data handling operations related to the decryption and encryption of the data assets that are utilised or produced in the process. Therefore, the Jobs Scheduler and Execution Engine is the key part of the ICARUS Application execution workflow as it orchestrates all the involved components in order to provide all the functionalities that compose the Application Execution Service as presented in deliverable D3.2.

During the ICARUS application execution, the Jobs Scheduler and Execution Engine receives the request for the execution or scheduled execution of the selected ICARUS application from the Analytics and Visualisation Workbench via its dedicated interface. This request triggers a series of actions towards the successful execution of the application. At first, a request is performed to the Resource Orchestrator in order to deploy the local Spark workers of the nodes of the Execution Cluster. Following the deployment preparation, the decryption of the data assets that will be used in the application's execution is performed with the help of the Decryption Manager. Once both tasks are finished, the Jobs Scheduler and Execution Engine performs the application execution. Once the execution is completed, the Jobs Scheduler and Execution Engine is interacting with the Encryption manager in order to encrypt the results and with the SecureSpace Worker in order to transfer them to the Core ICARUS platform so as to be stored. Once all operations have finished, the Analytics and Visualisation Workbench is notified.

As described above, the Jobs Scheduler and Execution Engine is exploiting the interfaces of the Resource Orchestrator, the Encryption and Decryption Manager, while also offering a set of the interfaces that are exploited by the Analytics and Visualization Workbench. The details of these interfaces of the Jobs Scheduler and Execution Engine, are presented in the following sub-section.

3.20.2 Interfaces

In the following tables the interfaces of the Jobs Scheduler and Execution Engine are defined.

Table 3-91: Jobs Scheduler and Execution Engine – get entry

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | JSEE01 |
| Endpoint Name | Get Entry |
| Endpoint Description | The service allows to access the entries from the scheduling table. |
| Component | Job Scheduler and Execution Engine |
| Endpoint URL | http://job-scheduler:8080/entries/{id} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> id : The id of the application that has to be scheduled |
| Request Body | N/A |
| Response Body | <pre>{ "entry": { { "id": long, "name": "string", "application_id": integer, "user_id": integer, "timezone": "string", "recurrent": boolean, "cron_exp": "string", "created_time": date, "updated_time": date } } }</pre> |

Table 3-92: Jobs Scheduler and Execution Engine - update entry

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | JSEE02 |
| Endpoint Name | Update entry |
| Endpoint Description | The service allows to manipulate the entries from the scheduling table. |
| Component | Job Scheduler and Execution Engine |
| Endpoint URL | http://job-scheduler:8080/entries/{id} |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> id: The id of the application that has to be scheduled |
| Request Body | <pre>{ "name": "string", "application_id": long, "user_id": integer, "timezone": "string", "recurrent": boolean, "cron_exp": "string", "start_time": date, "end_time": date }</pre> |

| | |
|----------------------|---|
| | } |
| Response Body | <pre>{ "entry": { { "id": long, "name": "string", "application_id": integer, "user_id": integer, "timezone": "string", "recurrent": boolean, "cron_exp": "string", "created_time": date, "updated_time": date } } }</pre> |

Table 3-93: Jobs Scheduler and Execution Engine - delete entry

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | JSEE03 |
| Endpoint Name | Delete entry |
| Endpoint Description | The service allows to delete the entry from the scheduling table. |
| Component | Job Scheduler and Execution Engine |
| Endpoint URL | http://job-scheduler:8080/entries/{id} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> id: The id of the application that has to be scheduled |
| Request Body | N/A |
| Response Body | <ul style="list-style-type: none"> 200 OK 401 Unauthorised 403 Forbidden 404 Not Found |

Table 3-94: Jobs Scheduler and Execution Engine - add entry

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | JSEE04 |
| Endpoint Name | Add new entry |
| Endpoint Description | The service allows to add new entry in the scheduling table. |
| Component | Job Scheduler and Execution Engine |

| | |
|---------------------------|---|
| Endpoint URL | http://job-scheduler:8080/entries |
| HTTP method | POST |
| Request Parameters | N/A |
| Request Body | <pre>{ "name": "string", "application_id": long, "user_id": long, "timezone": "string", "recurrent": boolean, "cron_exp": "string", "start_time": date, "end_time": date }</pre> |
| Response Body | <pre>{ "entry": { { "id": long, "name": "string", "application_id": long, "user_id": long, "timezone": "string", "recurrent": boolean, "cron_exp": "string", "created_time": date, "updated_time": date } } }</pre> |

Table 3-95: Jobs Scheduler and Execution Engine - get all entries

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | JSEE05 |
| Endpoint Name | Get all entries |
| Endpoint Description | The service allows to list the scheduling table. |
| Component | Job Scheduler and Execution Engine |
| Endpoint URL | http://job-scheduler:8080/entries |
| HTTP method | GET |
| Request Parameters | N/A |
| Request Body | N/A |
| Response Body | <pre>[{ "id": long, "name": "string", "application_id": long, "user_id": long, "recurrent": boolean, "cron_exp": "string", "created_time": date, "updated_time": date }]</pre> |

| | |
|--|--------|
| | }, ... |
| |] |

3.21 Notification Manager

3.21.1 Services Outline

In ICARUS platform, the Notification Manager is responsible for notifying users for any updates related to the data assets and the scheduled analytics jobs and its implementation follows the publish-subscribe pattern. As described in deliverable D3.2, the Notification Manager is involved in the Notifications service and is a key part of the Data Recommendation Workflows. Additionally, as described also in deliverable D3.2, each event type triggers a specific process that the Notification Manager follows in order to collect the information needed by interacting with the appropriate components and to notify the users. The exchange of data is realized using various events that are pushed in a message queue by the corresponding component. The requests of the Notification Manager for collecting further information are facilitated via REST APIs, provided by related components. All the notifications are shown in the notifications panel of the platform and the users can be notified using various communication channels such as emails, if the users select these types of notifications in their profiles. In order to provide all notification history to the users, the Notification Manager needs to maintain a notifications storage to store the notifications for each user in the ICARUS Storage.

When a data asset is added to ICARUS, a “DATASET_ADDITION” event is published. This event is published by the Data Handler which is responsible to provide the information related to the new data asset such as the data asset’s ID and its owner. Then, the Notification Manager which acts as a consumer of the events, will request from the ICARUS Storage component, via the Data Handler, the categories of this data asset and then, the users that have set similar categories as their preferences. Subsequently, the Notification Manager will store the notification in the ICARUS Storage, via the Data Handler, and notify these users via web notification and email.

In a similar manner, the Data Handler will publish a “DATASET_UPDATE” event for any update on an existing dataset. In this case, the message will contain the same information as the previous event, but the Notification Manager needs to request the list of users that are entitled to use this data asset from the Data License and Agreement Manager. Subsequently, the Notification Manager requests the users’ information from the ICARUS Storage, via the Data Handler, in order to notify them. Subsequently, the notification will be stored in the ICARUS Storage, again via the Data Handler.

The Data License and Agreement Manager is responsible to publish another event which is about a data asset request from a data consumer, called “DATASET_REQUEST”. This event contains information about the user that requested the data asset, as well as the data asset and its owner. At first, the Notification Manager is responsible to notify the data asset owner, by requesting the owner’s information from the ICARUS Storage, via the Data Handler, storing the notification to ICARUS Storage and notifying the owner. When the data owner approves or rejects the request, then the Data License and Agreement Manager will publish another event for this purpose, called “DATASET_RESPONSE”. In this case, Notification Manager will retrieve the information of the user that requested the data asset from the ICARUS Storage, via the Data Handler, store the notification in ICARUS Storage and notify the user that requested the data asset. The Notification Manager will generally provide notifications to the data provider and the data consumer in all steps during the contract preparation phase (regarding the CONTRACT_DRAFTED, CONTRACT_SIGNED, CONTRACT_REJECTED and CONTRACT_PAID events).

Finally, when there is an update on the execution status of a scheduled analytics job of a user, another event is published which is named “JOB_STATUS_UPDATE”. This event is generated by the Analytics and Visualization Workbench, containing information about the user ID, the analytics job ID and the new status (i.e. initiated, completed, failed) with any other relative information. This information will be used from the Notification Manager, so as to retrieve the information of the user who initiated the analytics job from ICARUS Storage, via the Data Handler, notify this user and store the notification in ICARUS Storage.

The details of the interfaces described above are presented in the following sub-section.

3.21.2 Interfaces

In the following tables the interfaces of the Notification Manager are defined. The first five interfaces are RESTful interfaces and the remaining five interfaces are related to the underlying publish / subscribe message queue. For the latter interfaces, the corresponding broker URL and the appropriate event message format are documented.

Table 3-96: Notification Manager - list retrieval

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | NM01 |
| Endpoint Name | Notifications list retrieval |
| Endpoint Description | Returns a list of notifications for a given user in a paginated format |
| Component | Notification Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/notifications/user/:uid?page=0&size=10&sort=created&order=dsc |

| | |
|---------------------------|---|
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • uid: user id • page: page index of paginated results (optional, default = 0) • size: number of maximum notification records to show on current page (optional, default = 10, max = 100) • sort: name of field to sort the results by (optional, default = createdAt) • order: ascending or descending sorting order of results ('asc' or 'dsc') |
| Request Body | N/A |
| Response Body | <p>The response body consists of a list of notifications records and pagination meta information in a JSON format. The notification list includes the notification's id, title, body, type of notification, an optional icon from the font awesome library, a URL to redirect the user on click and when the notification was created and seen. The pagination information includes the current page index, the number of notifications in the current page and in total, as well as the sorting field and order. For example:</p> <pre>{ notifications: [{ id: "5cff7698b399d52f54eea54a", title: "Data Asset Update", body: "Body text goes here...", type: "info" "warning" "success" "failure", icon: "fas fa-exclamation-circle", url: "https://hostname/..." seenAt: "2019-06-01T09:30:00.000Z", createdAt: "2019-06-01T09:30:00.000Z" }], { ... }}, page: 0, size: 2, sort: "createdAt", order: "dsc" "asc" total: 5 }</pre> |

Table 3-97: Notification Manager - single notification retrieval

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM02 |
| Endpoint Name | Single notification record retrieval |
| Endpoint Description | Returns a single notification record by the given notification id |
| Component | Notification Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/notifications/{nid} |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> • nid: notification id |
| Request Body | N/A |

| | |
|----------------------|--|
| Response Body | <p>The response body consists of a single notification record with the same information described in NM01. For example:</p> <pre>{ id: "5cff7698b399d52f54eea54a", title: "Data Asset Update", body: "Body text goes here...", type: "info" "warning" "success" "failure", icon: "fas fa-exclamation-circle", url: " https://hostname/..." seenAt: "2019-06-01T09:30:00.000Z", createdAt: "2019-06-01T09:30:00.000Z" }</pre> |
|----------------------|--|

Table 3-98: Notification Manager - mark notification as seen

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | NM03 |
| Endpoint Name | Mark notification as seen |
| Endpoint Description | Updates the given notification as seen and returns the newly updated notification |
| Component | Notification Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/notifications/{nid}/seen |
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> nid: notification id |
| Request Body | N/A |
| Response Body | <p>The response body consists of the newly updated notification record. For example:</p> <pre>{ id: "5cff7698b399d52f54eea54a", title: "Data Asset Update", body: "Body text goes here...", type: "info" "warning" "success" "failure", icon: "fas fa-exclamation-circle", url: " https://hostname/..." seenAt: "2019-06-01T09:30:00.000Z", createdAt: "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-99: Notification Manager - mark all notifications as seen

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | NM04 |
| Endpoint Name | Mark all notifications as seen |
| Endpoint Description | Updates all the notification of a given user as seen |
| Component | Notification Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/notifications/{uid} |

| | |
|---------------------------|--|
| HTTP method | PUT |
| Request Parameters | <ul style="list-style-type: none"> uid: user id |
| Request Body | N/A |
| Response Body | N/A |

Table 3-100: Notification Manager - delete notification

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM05 |
| Endpoint Name | Delete Notification |
| Endpoint Description | Removes the notification from storage |
| Component | Notification Manager |
| Endpoint URL | https://icarus_platform[:port]/api/v1/notifications/{nid} |
| HTTP method | DELETE |
| Request Parameters | <ul style="list-style-type: none"> nid: notification id |
| Request Body | N/A |
| Response Body | N/A |

Table 3-101: : Notification Manager - new data asset notification

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM06 |
| Event Name | New data asset notification |
| Event Description | Event for notifying interested users for the addition of a new data asset |
| Component | Notification Manager |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_notifications/ |
| Event Message | <p>The event message should include the newly added data asset id and data asset owner's id. For example:</p> <pre>{ "event_type": "DATASET_ADDITION", "dataAssetId": "D12345678", "dataAssetOwnerId": "U12345678" }</pre> |

Table 3-102: Notification Manager - data asset update notification

| ICARUS Technical Interface | |
|-------------------------------|------|
| Technical Interface ID | NM07 |

| | |
|--------------------------|--|
| Event Name | Data asset update notification |
| Event Description | Event for notifying interested users for updates on data assets |
| Component | Notification Manager |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_notifications/ |
| Event Message | <p>The event message should include the updated data asset id and data asset owner's id. For example:</p> <pre>{ "event_type": "DATASET_UPDATE", "dataAssetId": "D12345678", "dataAssetOwnerId": "U12345678" }</pre> |

Table 3-103: Notification Manager - data asset request notification

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM08 |
| Event Name | Data asset request notification |
| Event Description | Event for notifying a data asset owner for access requests from users |
| Component | Notification Manager |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_notifications/ |
| Event Message | <p>The event message should include the requested data asset id, owner and requester id. For example:</p> <pre>{ "event_type": "DATASET_REQUEST", "dataAssetId": "D12345678", "dataAssetOwnerId": "U12345678", "dataAssetRequesterId": "U12344444" }</pre> |

Table 3-104: Notification Manager - data asset draft contract notification

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM09 |
| Event Name | Data asset response notification for a draft contract ready for signature |
| Event Description | Event for notifying a data asset requester for the owners' response |
| Component | Notification Manager |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_notifications/ |
| Event Message | <p>The event message should include the requested data asset id, owner and requester id. For example:</p> <pre>{ "event_type": "DATASET_RESPONSE", "dataAssetId": "D12345678", "dataAssetOwnerId": "U12345678", "dataAssetRequesterId": "U12344444", "contactId": "XYZ12345678", }</pre> |

| | |
|--|---|
| | } |
|--|---|

Table 3-105: Notification Manager – job status update notification

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | NM10 |
| Event Name | Job status update notification |
| Event Description | Event for notifying a user for any update on the execution status of his scheduled analytics job |
| Component | Notification Manager |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_notifications/ |
| Event Message | <p>The event message should include the analytics job id, the user's id and the job's status. For example:</p> <pre>{ "event_type": "JOB_STATUS_UPDATE", "jobId": "J12345678", "userId": "U12345678", "status": "initiated" "completed" "failed" }</pre> |

3.22 Usage Analytics

3.22.1 Services Outline

Usage Analytics is the component of the ICARUS architecture that is responsible for providing meaningful platform utilization insights by collecting, aggregating and visualizing statistics related to the usage of data (e.g. total number of purchases) and service assets (e.g. total number of times an application was used), as well as usage statistics of the platform (e.g. total number of active users) and the users' private space (e.g. number of active analytics jobs). As described in deliverable D3.2, the implementation of Usage Analytics is based on the publish-subscribe pattern and consumes the events that various components publish. Using these events, the Usage Analytics component is able to interact with the various ICARUS components and receive all the needed inputs.

The Policy Manager is responsible to generate events every time a user registers to the ICARUS platform and every time the user logs in/out to/from the platform. These events, called "USER_REGISTER", "USER_LOGIN" and "USER_LOGOUT", contain information such as the user ID and the specific timestamp of the related action. Additional statistics will be aggregated at organization level.

The Query Explorer is responsible to generate events for the exploration of the assets. This involves information related to which assets appear in search results included in "ASSETS_APPEARED" events, as well as which of them are viewed, starred and unstarred in

“ASSET_VIEWED”, “ASSET_STARRED” and “ASSET_UNSTARRED” events, respectively. These events include the asset ID, the related user ID and the specific timestamp in each case.

The Data License and Agreement Manager is responsible to publish “ASSET_REQUESTED”, “ASSET_REQUEST_REJECTED”, “CONTRACT_DRAFTED”, “CONTRACT_SIGNED”, “CONTRACT_REJECTED” and “ASSET_PURCHASED” / “CONTRACT_PAID” events for assets requests and contracts. These types of events will contain information such as the asset ID, the asset consumer who requested/purchased the asset, the asset’s provider, and the timestamp of the event.

The Data Handler is responsible to generate an event “ASSET_ADDITION” when an asset is added to ICARUS and an event “ASSET_UPDATED” when an existing asset is updated in the ICARUS platform. Such events are published by the Data Handler which is responsible to provide the information related to the new asset such as the asset’s ID and its owner, as well as the categories/tags of the asset.

Furthermore, the “ALGORITHM_USED” and “VISUALIZATION_USED” events that include information related to the usage of the algorithms and the visualizations are published by the Analytics and Visualization Workbench. This information is about the usage of each algorithm and visualization, as well as the user that utilized the application, including the specific timestamp. The Analytics and Visualization Workbench also publishes events about the users’ analytics jobs, including information about their jobs’ status and timestamps. “JOB_STARTED” event is published when an analytics job is initiated, “JOB_SUCCESS” event is generated when an analytics job is successfully completed and “JOB_FAILED” event is published when an analytics job has stopped due to an error (e.g. lack of resources).

In addition, the Resource Orchestrator publishes events for the usage of the platform’s computational resources. Specifically, it publishes “VM_STARTED” and “VM_STOPPED” events, for the allocation and de-allocation of computational resources (e.g. CPU, memory, storage, etc.) to the users’ private spaces respectively, accompanied with the specific timestamps.

The Usage analytics component consumes these events and stores them in the ICARUS Storage via the Data Handler. When a user requires to visualize the various statistics, Usage Analytics interacts with the ICARUS Storage, via the Data Handler, and retrieves, aggregates and visualizes the statistics to the user.

The details of the interfaces described above are presented in the following sub-section.

3.22.2 Interfaces

As described in the previous sub-section, the Usage Analytics component is following a publish subscribe pattern utilising a message queue. Hence, in the following tables the

interfaces of the Usage Analytics are defined, documenting the corresponding broker URL and the appropriate event message format.

Table 3-106: Usage Analytics - user registration

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | UA01 |
| Event Name | User registration |
| Event Description | Stores the registration event of a user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the newly registered user and the datetime of the user registration. For example:</p> <pre>{ "event_type": "USER_REGISTER", "user_id": "U12345678", "registered_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-107: Usage Analytics - user login

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | UA02 |
| Event Name | User log in |
| Event Description | Stores the login event of a user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the logged in user and the datetime of the login. For example:</p> <pre>{ "event_type": "USER_LOGIN", "user_id": "U12345678", "login_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-108: Usage Analytics - user logout

| ICARUS Technical Interface | |
|----------------------------|------------------------------------|
| Technical Interface ID | UA03 |
| Event Name | User log out |
| Event Description | Stores the logout event of a user. |

| | |
|----------------------|--|
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the logged-out user and the datetime of the logout. For example:</p> <pre>{ "event_type": "USER_LOGOUT", "user_id": "U12345678", "logout_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-109: Usage Analytics - asset appeared in search

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | UA04 |
| Event Name | Assets appeared in search results |
| Event Description | Stores a list of assets ids that appeared in the search results of a user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the ids of the assets that appeared from the query results, the user id and the datetime. For example:</p> <pre>{ "event_type": "ASSETS_APPEARED", "assets_ids": ["D11111111", "D22222222", ...], "user_id": "U12345678", "assets_appeared_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-110: Usage Analytics - asset viewed

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA05 |
| Event Name | Asset viewed |
| Event Description | Stores the event that an asset was viewed by a specific user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |

| | |
|----------------------|---|
| Event Message | <p>The event message should include the id of the asset that was viewed, the user id and the datetime. For example:</p> <pre>{ "event_type": "ASSET_VIEWED", "asset_id": "D11111111", "user_id": "U12345678", "viewed_at": "2019-06-01T09:30:00.000Z" }</pre> |
|----------------------|---|

Table 3-111: Usage Analytics - asset starred

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | UA06 |
| Event Name | Asset starred |
| Event Description | Stores the event that an asset was starred by a specific user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was starred, the user id and the datetime. For example:</p> <pre>{ "event_type": "ASSET_STARRED", "asset_id": "D11111111", "user_id": "U12345678", "starred_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-112: Usage Analytics - asset un-starred

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA07 |
| Event Name | Asset un-starred |
| Event Description | Stores the event that an asset was un-starred by a specific user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was un-starred, the user id and the datetime. For example:</p> <pre>{ "event_type": "ASSET_UNSTARRED", "asset_id": "D11111111", "user_id": "U12345678", "unstarred_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-113: Usage Analytics - asset requested

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | UA08 |
| Event Name | Asset requested |
| Event Description | Stores the event that an asset was requested by a specific user (asset consumer). |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was requested, the id of the user that requested the asset and the datetime. For example:</p> <pre>{ "event_type": "ASSET_REQUESTED", "asset_id": "D11111111", "user_id": "U12345678", "requested_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-114: Usage Analytics - asset request rejected

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | UA09 |
| Event Name | Asset request rejected |
| Event Description | Stores the event that a request for an asset was rejected by the asset owner. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was requested, the id of the user (consumer) that requested the asset, the id of the user (asset owner) that rejected the request and the datetime. For example:</p> <pre>{ "event_type": "ASSET_REQUEST_REJECTED", "asset_id": "D11111111", "asset_consumer_id": "U12345678", "asset_owner_id": "U22222222", "requested_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-115: Usage Analytics - asset purchased

| ICARUS Technical Interface | |
|----------------------------|-----------------|
| Technical Interface ID | UA10 |
| Event Name | Asset purchased |

| | |
|--------------------------|--|
| Event Description | Stores the event that an asset was purchased by the asset consumer. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was purchased, the id of the user (consumer) that purchased the asset, the id of the user (asset owner) that sold the asset, the date (start, end) that the contract will be active and the datetime of the purchase. For example:</p> <pre>{ "event_type": "ASSET_PURCHASED", "asset_id": "D11111111", "asset_consumer_id": "U12345678", "asset_owner_id": "U22222222", "contract_start_date": "2019-06-01T09:30:00.000Z", "contract_end_date": "2020-06-01T09:30:00.000Z", "purchased_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-116: Usage Analytics - asset created

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA11 |
| Event Name | Asset created/uploaded |
| Event Description | Stores the event that an asset was created or uploaded by an asset provider. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the asset that was uploaded/created, the id of the user (asset owner) that uploaded/created the asset, whether it is a data asset or a service asset, and the datetime that the asset was uploaded/created. For example:</p> <pre>{ "event_type": "ASSET_ADDITION", "asset_id": "D11111111", "asset_owner_id": "U22222222", "is_data_asset": "1", "created_uploaded_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-117: Usage Analytics - algorithm utilised

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA12 |
| Event Name | Algorithm utilized |
| Event Description | Stores the event that an algorithm was used by a specific user. |

| | |
|----------------------|--|
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the user that used the algorithm, the id of the algorithm and the datetime that the algorithm was used. For example:</p> <pre>{ "event_type": "ALGORITHM_USED", "user_id": "U12345678", "algorithm_id": "A12345678", "utilized_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-118: Usage Analytics - visualisation utilised

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | UA13 |
| Event Name | Visualization utilized |
| Event Description | Stores the event that a visualization was used by a specific user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the user that used the visualization, the id of the visualization and the datetime that the visualization was used. For example:</p> <pre>{ "event_type": "VISUALIZATION_USED", "user_id": "U12345678", "visualization_id": "V12345678", "utilized_at": "2019-06-01T09:30:00.000Z" }</pre> |

Table 3-119: Usage Analytics - vm started

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA14 |
| Event Name | VM started |
| Event Description | Stores the event of the allocation of a VM. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the new vm and the resources of the vm. For example:</p> <pre>{ "event_type": "VM_STARTED", "vm_id": "V12345678", "vm_resources": {</pre> |

| | |
|--|---|
| | <pre> "CPUs": "16", "RAM_GB": "32", "Disk_Bytes": "1073741824" }, "started_at": "2019-06-01T09:30:00.000Z" } </pre> |
|--|---|

Table 3-120: Usage Analytics - vm stopped

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | UA15 |
| Event Name | VM stopped |
| Event Description | Stores the event of the de-allocation of a VM. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the vm, the resources of the vm that was stopped, the total resources of the platform and the datetime that the vm stopped. For example:</p> <pre> { "event_type": "VM_STOPPED", "vm_id": "V12345678", "vm_resources": { "CPUs": "16", "RAM_GB": "32", "Disk_Bytes": "1073741824" }, "stopped_at": "2019-06-01T09:30:00.000Z" } </pre> |

Table 3-121: Usage Analytics - new analytics job

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | UA16 |
| Event Name | New analytic job |
| Event Description | Stores the event of the initialization of a new job from a user. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the user that started the new job, the id of the new job, the resources for the job, the available resources of the user after the new job, and the datetime that the job started. For example:</p> <pre> { "event_type": "JOB_STARTED", "user_id": "U12345678", "job_id": "J12345678", </pre> |

| | |
|--|--|
| | <pre> "job_resources": { "CPUs": "16", "RAM_GB": "32", "Disk_Bytes": "1073741824" }, "user_available_resources": { "CPUs": "160", "RAM_GB": "320", "Disk_Bytes": "107374182400" }, "started_at": "2019-06-01T09:30:00.000Z" } </pre> |
|--|--|

Table 3-122: Usage Analytics - analytics job success

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | UA17 |
| Event Name | Successfully completed analytic job |
| Event Description | Stores the event of the successful completion of a job. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the completed job, and the datetime that the job completed. For example:</p> <pre> { "event_type": "JOB_SUCCESS", "job_id": "J12345678", "completed_at": "2019-06-01T09:30:00.000Z" } </pre> |

Table 3-123: Usage Analytics - analytics job failed

| ICARUS Technical Interface | |
|-------------------------------|--|
| Technical Interface ID | UA18 |
| Event Name | Failed analytic job |
| Event Description | Stores the event of the unsuccessful completion of a job and the error of the failed job. |
| Component | Usage Analytics |
| Broker URL | https://icarus_platform[:port]/api/v1/kafka_usage_analytics/ |
| Event Message | <p>The event message should include the id of the job that failed, and the datetime that the job failed. For example:</p> <pre> { "event_type": "JOB_FAILED", "job_id": "J12345678", "error_id": "error-1", </pre> |

| | |
|--|--|
| | <pre> "failed_at": "2019-06-01T09:30:00.000Z" } </pre> |
|--|--|

Table 3-124: Usage Analytics - general assets statistics

| ICARUS Technical Interface | |
|----------------------------|---|
| Technical Interface ID | UA19 |
| Endpoint Name | General asset statistics |
| Endpoint Description | Returns aggregated statistics about total number of views, stars and purchases for a specific asset. |
| Component | Usage Analytics |
| Endpoint URL | https://icarus_platform[:port]/api/v1/usage-analytics/general_asset_statistics?asset_id=[asset_id] |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> asset_id: the unique id of an asset |
| Request Body | N/A |
| Response Body | <p>The response body consists of the total number of views, stars and purchases for a specific asset. For example:</p> <pre> { "number_of_views": "215", "number_of_stars": "35", "number_of_purchases": "8" } </pre> |

Table 3-125: Usage Analytics - private assets statistics

| ICARUS Technical Interface | |
|----------------------------|--|
| Technical Interface ID | UA20 |
| Endpoint Name | Private asset statistics |
| Endpoint Description | Returns several metrics/statistics for an asset. This endpoint can be accessed only by the asset owners of the specific asset. |
| Component | Usage Analytics |
| Endpoint URL | https://icarus_platform[:port]/api/v1/usage-analytics/private_asset_statistics?asset_id=[asset_id]&from=yyyy-MM-dd&to=yyyy-MM-dd |
| HTTP method | GET |

| | |
|---------------------------|--|
| Request Parameters | <ul style="list-style-type: none"> asset_id: the unique id of an asset from: returns statistics after the given date. Date should be formatted as yyyy-MM-dd. to: returns statistics before the given date. Date should be formatted as yyyy-MM-dd. |
| Request Body | N/A |
| Response Body | <p>The response body consists of several different private metrics for an asset that only the asset owner has access. For example:</p> <pre> { "number_of_views": "215", "views_per_time_period": [{ "date": "2019-06-01", "views": "5" }, { ... }], "number_of_stars": "35", "stars_per_time_period": [{ "date": "2019-06-01", "stars": "25" }, { ... }], "number_of_purchases": "8", "purchases_per_time_period": [{ "date": "2019-06-01", "purchases": "2" }, { ... }], "number_of_requests_received": "20", "requests_per_time_period": [{ "date": "2019-06-01", "requests": "10" }, { ... }], "number_of_rejected_requests": "10", "rejected_requests_per_time_period": [{ "date": "2019-06-01", "rejected_requests": "5" }, { ... }] } </pre> |

Table 3-126: Usage Analytics - user private statistics

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA21 |
| Endpoint Name | Organization private statistics |
| Endpoint Description | Returns several metrics/statistics for the profile and private space of an organization. |
| Component | Usage Analytics |
| Endpoint URL | https://icarus_platform[:port]/api/v1/usage-analytics/user_private_statistics?user_id=[user_id]&from=yyyy-MM-dd&to=yyyy-MM-dd |

| | |
|---------------------------|--|
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> organization_id: the unique id of an organization from: returns statistics after the given date. Date should be formatted as yyyy-MM-dd. to: returns statistics before the given date. Date should be formatted as yyyy-MM-dd. |
| Request Body | N/A |
| Response Body | <p>The response body consists of several different private metrics for an organization. For example:</p> <pre> { "user_current_available_resources": { "CPUs": "160", "RAM_GB": "320", "Disk_Bytes": "107374182400" }, "user_available_resources_per_time_period": [{ "date": "2019-07-01", "resources": { "CPUs": "160", "RAM_GB": "320", "Disk_Bytes": "107374182400" } }, { ... }], "number_of_current_active_jobs": "10", "active_jobs_per_time_period": [{ "date": "2019-07-01", "active_jobs": "5" }, { ... }], "completed_jobs_per_time_period": [{ "date": "2019-07-01", "completed_jobs": "5" }, { ... }], "failed_jobs_per_time_period": [{ "date": "2019-07-01", "failed_jobs": "5" }, { ... }], "number_of_requests": "10", "requests_per_time_period": [{ "date": "2019-07-01", "requests": "5" }, { ... }], "number_of_data_assets_uploaded": "10", "data_assets_uploaded_per_time_period": [{ "date": "2019-07-01", "uploaded_data_assets": "5" }, { ... }], "number_of_service_assets_created": "10", "service_assets_created_per_time_period": [{ "date": "2019-07-01", "created_service_assets": "5" }, { ... }] } </pre> |

Table 3-127: Usage Analytics - admin private statistics

| ICARUS Technical Interface | |
|-------------------------------|---|
| Technical Interface ID | UA22 |
| Endpoint Name | Admin private statistics |
| Endpoint Description | Returns several aggregated metrics/statistics for the admin user. This endpoint can be accessed only by the admin. |
| Component | Usage Analytics |
| Endpoint URL | https://icarus_platform[:port]/api/v1/usage-analytics/admin_private_statistics?from=yyyy-MM-dd&to= yyyy-MM-dd |
| HTTP method | GET |
| Request Parameters | <ul style="list-style-type: none"> from: returns statistics after the given date. Date should be formatted as yyyy-MM-dd. to: returns statistics before the given date. Date should be formatted as yyyy-MM-dd. |
| Request Body | N/A |
| Response Body | <p>The response body consists of several different private metrics for the admin. For example:</p> <pre> "total_registered_users": "150", "number_of_new_users_per_time_period": [{ "date": "2019-07-01", "number_of_users": "10" },{...}], "number_of_active_users": "25", "number_of_active_users_per_time_period": [{ "date": "2019-07-01", "number_of_active_users": "10" },{...}], "total_uploaded_data_assets": "85", "number_of_uploaded_data_assets_per_time_period": [{ "date": "2019-07-01", "number_of_data_assets": "10" },{...}], "total_created_applications": "15", "number_of_created_applications_per_time_period": [{ "date": "2019-07-01", "number_of_applications": "10" },{...}], "total_asset_requests": 50, "number_of_asset_requests_per_time_period": [{ "date": "2019-07-01", "number_of_requests": "10" },{...}], "total_rejected_requests": "20", "number_of_rejected_requests_per_time_period": [{ "date": "2019-07-01", "number_of_rejected_requests": "10" } </pre> |

```

    },{...}
  ],
  "total_purchased_assets": "30",
  "number_of_purchased_assets_per_time_period": [{
    "date": "2019-07-01", "number_of_purchases": "10"
  },{...}
  ],
  "popularity_of_analytics_algorithms": [{
    "algorithm": "k-means", "number_of_times_used": "10"
  },{...}
  ],
  "popularity_of_visualizations": [{
    "visualization": "bar-chart", "number_of_times_used": "10"
  },{...}
  ],
  "platform_available_resources": {
    "CPUs": "160",
    "RAM_GB": "320",
    "Disk_Bytes": "107374182400"
  },
  "platform_available_resources_per_time_period": [{
    "date": "2019-07-01",
    "resources": {
      "CPUs": "160",
      "RAM_GB": "320",
      "Disk_Bytes": "107374182400"
    }
  }, { ... }
  ],
  "number_of_current_active_jobs": "10",
  "active_jobs_per_time_period": [{
    "date": "2019-07-01", "active_jobs": "5"
  }, { ... }
  ],
  "completed_jobs_per_time_period": [{
    "date": "2019-07-01", "completed_jobs": "5"
  }, { ... }
  ],
  "failed_jobs_per_time_period": [{
    "date": "2019-07-01", "failed_jobs": "5"
  }, { ... }
  ]

```

4 Conclusions & Next Steps

The purpose of this deliverable entitled D3.3 “Architecture, Core Data and Value Added Services Bundles Specifications-v2.00” was to deliver the complementary documentation, updating the information documented in deliverable D3.1, for the ICARUS platform architecture, as well as for the core functionalities and the interfaces of the components that compose the ICARUS platform. The deliverable is built directly on top of the main outcomes and the knowledge extracted from deliverables D3.1 and D3.2 in order to deliver the required updates and refinements on the documentation of the components of the platform, as a result of the thorough analysis that was performed on the ICARUS platform’s workflows and the design of the services of the platform.

At first, the architecture of the integrated ICARUS platform is further elaborated, providing the necessary description of the updated components of the platform focusing on their positioning within platform, the functionalities of the platform that they are involved in and their interactions with the other components in order to deliver these functionalities.

Following the complementary documentation of the ICARUS architecture, the necessary updates on the documentation of the ICARUS components, that was presented in deliverable D3.1, are documented. For each component of the platform, the core functionalities that they offer are described. Furthermore, their involvement in the ICARUS platform’s workflows and the services is presented, focusing on the communication and interactions with the rest of the components for the implementation of these workflows and services. To facilitate these interactions, a set of interfaces is described for each component that enable the required exchange of information between the components. Furthermore, the complete technical details of these interfaces are documented. In total, 127 interfaces were documented.

The current deliverable presented the updated documentation of the architecture of the integrated ICARUS platform and of all the components of the platform. However, as the project evolves and the development activities are progressing, the design of the ICARUS platform’s architecture and of the components of the architecture will receive the necessary updates and optimisations in order to encapsulate all the project’s advancements, as well as the new technical requirements that will be extracted from the feedback that will be collected from the platform’s development and evaluation activities. Hence, the forthcoming versions of this deliverable will incorporate all the updates that are necessary to be introduced.