DE4A IAL Technical Design v1.1

Author: Philip Helger, philip@helger.com, Version: 2022-04-01

1 Introduction

This document describes the technical design of the DE4A IAL.

It is based on the mock IAL of Iteration 1 and provides an extended design for Iteration 2, also with an extended API.

1.1 References

[BC-XSD] Business Card XML Schema, https://docs.peppol.eu/edelivery/directory/peppol-directory-business-card-20180621.xsd

[Codelists] DE4A Code Lists, https://github.com/de4a-wp5/de4a-codelists

[IAL-XSD] IAL XML Schema specification, https://github.com/de4a-wp5/ial-service/tree/main/ial-api/src/main/resources/schemas

[PD] Peppol Directory specification, v1.1.1, https://docs.peppol.eu/edelivery/directory/PEPPOL-EDN-Directory-1.1.1-2020-10-15.pdf

1.2 Out of scope

This document explicitly does not deal with:

- The handling of the MOR (Multilingual Ontology Repository)

2 Public API

This chapter describes the public API of the IAL. The XML schema with the required data structures can be found at [IAL].

Compared to the original WP3 design the public API solely relies on the use of XML for communication. Hence no OpenAPI or similar document is available.

2.1 Data Owner queries

This chapter groups all the queries that can be used to query Data Owners (DOs). This functionality is called DSD (Data Services Directory) in TOOP and SDG-OOP.

2.1.1 /provision/{canonicalObjectTypeIDs}[/{atuCode}] API

Get all Data Owners that support the requested canonical object types. If an administrative territorial unit (ATU) is provided, it will also be taken into account.

Note: URLs are limited to 2048 characters. Therefore, only a maximum of 20 canonical object type IDs are safely supported.

Note: this API provides no special handling for the multi evidence case. It will only return the canonical evidence object types it was queried for.

Compatibility note: this interface replaces the following interfaces present in Iteration 1:

`/ial/{canonicalEvidenceTypeId}`

`/ial/{canonicalEvidenceTypeId}/{countryCode}`

The old APIs allowed only one canonical evidence type ID and optionally only a country code. This is a breaking change but can easily be hidden from implementers by keeping the Connector query API unchanged.

URL parameters:

- `{canonicalObjectTypeIDs}` – mandatory – one or more canonical object type identifiers, including the identifier schemes. Multiple IDs need to be separated by the comma (,) character. A maximum of 20 different values is allowed.
- `{atuCode}` – optional – an administrative territorial unit (ATU) code on an arbitrary level. This can be a country code, a NUTS 1, a NUTS 2, a NUTS 3 or a LAU code. If no ATU code is provided, results from all regions will be returned.

Query parameters:

- none

Request body:

- none

Response body:

- An XML document of type `ResponseLookupRoutingInformationType` from IAL.xsd. If nothing is found, this is indicated in this response. If the `Accept` HTTP header favours “application/json” over “application/xml”, a JSON representation will be returned.

Status codes:

- HTTP 200 Success
  - If the message was processed correctly
- HTTP 400 Bad Request
  - If the request payload does not match the requirements (e.g. invalid parameters)
- HTTP 500 Internal Server Error
  - If something unexpected happened

Examples calls (non-normative):

The examples are displayed without URL encoding. In reality e.g. all “:” characters must be replaced with “%3A” to work.

- `/provision/urn:de4a-eu:CanonicalEvidenceType::CompanyRegistration:1.0`
  - Search for all EPs that support the “Company Registration” evidence type, independent of the country
- `/provision/urn:de4a-eu:CanonicalEvidenceType::CompanyRegistration:1.0/AT`
  - Search for all EPs that support the “Company Registration” evidence type, limit to the matches in Austria
- `/provision/urn:de4a-eu:CanonicalEvidenceType::CompanyRegistration:1.0/AT130`
  - Search for all EPs that support the “Company Registration” evidence type, limit to the matches in Vienna, Austria (NUTS 3)

- `/provision/urn:de4a-eu:CanonicalEvidenceType::MarriageRegistration:1.0,urn:de4a-eu:CanonicalEvidenceType::BirthCertificate:1.0`
  - Search for all EPs that support the “Marriage Registration” or the “Birth Certificate” evidence type, independent of the country
- `/provision/urn:de4a-eu:CanonicalEvidenceType::MarriageRegistration:1.0,urn:de4a-eu:CanonicalEvidenceType::BirthCertificate:1.0/SE`
  - Search for all EPs that support the “Marriage Registration” or the “Birth Certificate” evidence type, limit to the matches in Sweden

3 Infrastructure setup

3.1 Big Picture IAL based on Directory

This chapter describes the big picture of the IAL implementation, based on Business Cards provided in a decentralized way through the DE4A Directory.

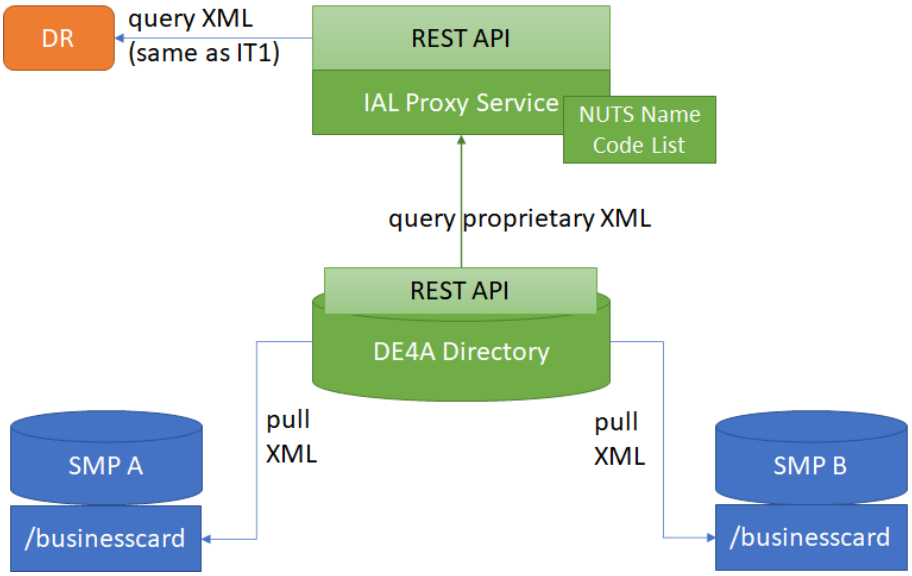

Figure 1: System overview

The above figure shows the relationships of the different components. The blue SMP instances are already existing and are representing pilot partners SMP instances. These SMPs need to provide Business Cards (see chapter 3.2) to a centralized Directory (see chapter 3.3). This Directory is an existing piece of software (based on the original Peppol Directory) which offers a standardized REST API for querying. This REST API will be used by the IAL Proxy Service (see chapter 3.4), which in turn offers the public API as described in chapter 1.2 to any DR.

The benefit of this setup is, that pilots can maintain the data decentralized in their SMPs. This data is automatically copied (replicated) into the central Directory where a complete index of all Business Cards from all SMPs is available for centralized querying. The Directory offers only a read-only API to non-SMPs, so there can never be a data update from the Directory to an SMP.

3.2 Business Card

3.2.1 General information

The Business Card represents additional metadata for a single DE4A Participant ID. The data model is specified in the Peppol Directory specification and not customizable. See [BC-XSD] for details. The Business Card is only a data model, with a defined XML representation.

The data model needs to be filled by the Data Owners (C4) and provided by the Data Transferors (C3) in the matching SMPs. The interface for providing the Business Cards is defined in [PD] and ready to be used in all "phoss SMP" instances. By previously limiting the usage of SMPs to phoss SMP instances, only minor configuration work needs to be done by pilot partners.

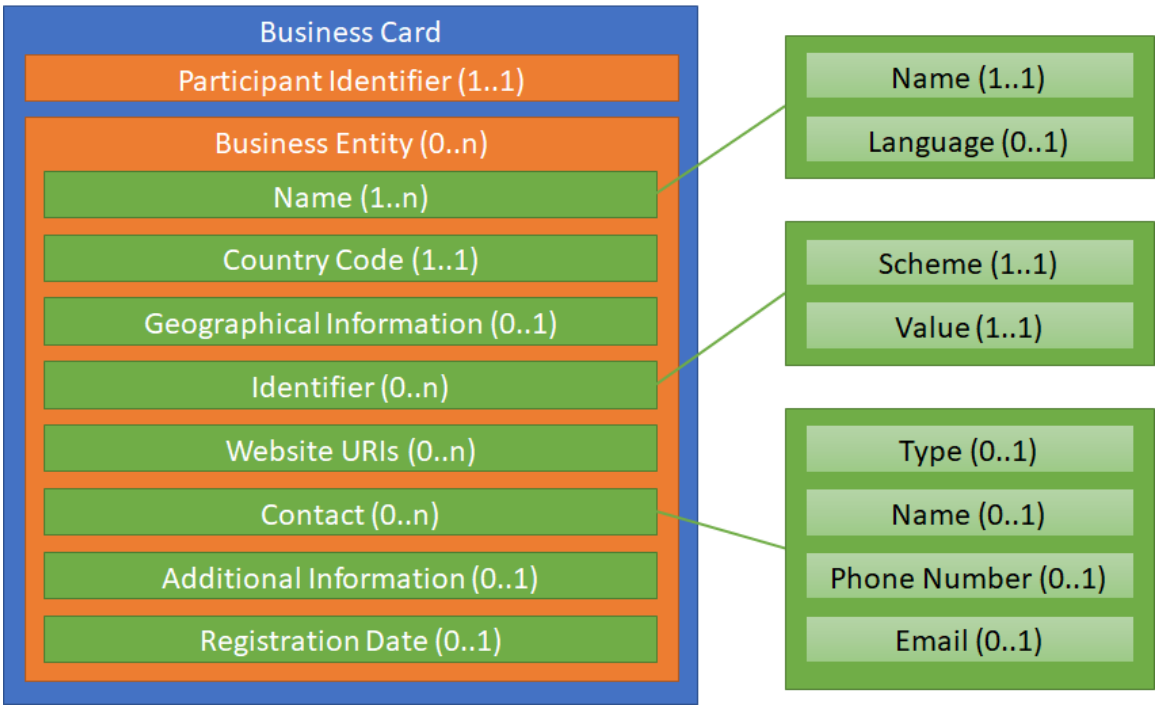

Figure 2: Business Card data model

3.2.2 IAL mapping

The fields of the Business Card need to be mapped to the result structure of the IAL, as defined in [IAL]. See the following table for a mapping from the Business Card XML to the currently defined draft JSON. This mapping may change for the final implementation.

Business Card (XML)	IAL response (JSON)	Cardinality	Comment
/BusinessCard/ParticipantIdentifier	/dataOwnerId	1..1	
/BusinessCard/BusinessEntity /Name/Name	/dataOwnerPrefLabel	1..1	
/BusinessCard/BusinessEntity /CountryCode	/atuCode	1..1	Fall
--	/atuLevel		Calculated from ATU code based on the layout
/BusinessCard/BusinessEntity /Identifier[scheme='atuCode']	/atuCode	0..1	Has preference over CountryCode
--	/atuLatinName		Calculated from ATU code or Country Code

			based on a mapping table
/BusinessCard/BusinessEntity /AdditionalInformation	/parameterSetList	0..1	Additional parameters for querying must be provided in the JSON specific syntax.

As “ParticipantIdentifier”, “Name” and “CountryCode” are technically required fields of each Business Card.

The ATU code needs to be mapped onto the `/BusinessCard/BusinessEntity/Identifier` element where the `scheme` attribute needs to have the value `atuCode` (case sensitive) and the text context of the element needs to contain the real ATU code (e.g. “AT130”, “SI” or “ES11”).

In some rare cases, it must be possible to also add a “parameter set”, which is a complex structured element. As a decomposition into a flat hierarchy would be very error prone, it was decided to force the technical syntax into the `AdditionalInformation` element. It MUST be a JSON array and the structure MAY look like this:

```
[
  {
    "title": "ES/BirthEvidence/BirthRegister",
    "parameterList": [
      {
        "name": "ES/Register/Volume",
        "optional": false
      }
    ]
  }
]
```

A respective XML representation will be provided by WP5.

How these technical details are made available to DE4A pilot partners, WP5 and the Playground operators is described below in chapter 4.

3.3 DE4A Directory

3.3.1 Functionality

The DE4A Directory is a specific instance of the “phoss Directory” originally created for the Peppol project as the “Peppol Directory” (see [PD]). The purpose of the Directory is to be the central place that collects all the Business Cards from all the SMPs and offers a search, both for humans as well as for machines (via a REST API).

It’s the responsibility of each SMP to inform the Directory about any relevant change. This is done by a predefined API offered by the Directory and invoked by an SMP. This API pushes the Participant Identifier which was changed to the Directory. Then the Directory resolves the owning SMP of the Participant Identifier via the DNS system, to ensure that the Business Card is only taken from the trustworthy owner SMP and not from “any” SMP.

All the Business Cards of all SMPs are collectively indexed in a Lucene index. Apache Lucene[1] is a high-performance, full-featured search engine library written entirely in Java. This index can be queried via a web UI or a REST API. Further details can be found in the [PD] specification.

The following figure depicts the flow of the communication. The left side of the image focuses on the filling of the Directory, whereas the right part focuses on the querying and searching.

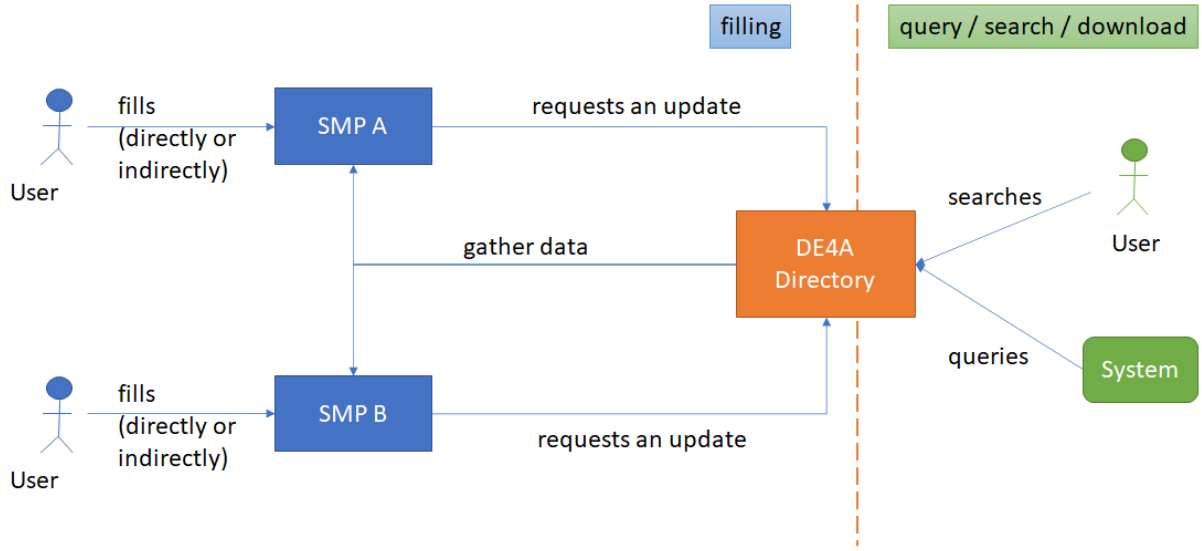

Figure 3: Directory big picture

3.3.2 Software

The software to run the DE4A Directory is the "phoss Directory" software. It's an Open-Source software, licensed under the Apache 2.0 license. It's hosted on https://github.com/phax/phoss-directory.

3.3.3 Operations

The DE4A Directory needs to be set up once by the project and hosted centrally. The DE4A Directory must be accessible via the Internet and offer only secured access via TLS. A specific configuration file needs to be provided, so that the DNS connectivity can be established.

The Directory is an open service that does not require authentication. As it operates solely with open data, no additional security measures needed to be provided.

To avoid being flooded by queries, the Directory supports an internal API rate limit mechanism.

3.4 IAL Proxy Service

3.4.1 Functionality

The IAL Proxy Service is a simple service which the project refers to as "IAL". Its goal is to get queried by any DR to find matching DOs that fulfil certain criteria like association to a specific ATU or support for a specific document type.

The IAL Proxy Service must implement the API as described in chapter 2.1. It decomposes the request and converts it into one or more DE4A Directory API calls. The results of the Directory API calls need to be aggregated, enriched and synchronously returned to the caller. So, the IAL Proxy Service is considered a stateless service.

[1] See https://lucene.apache.org/

Enrichment particularly includes the determination of the ATU level as well as the Latin name of the ATU code to be returned. The ATU level can be determined via an algorithm from the ATU code and the Latin name of an ATU code can be taken from a table[2] offered by the EC.

3.4.2 Implementation

The IAL Proxy Service will be implemented by WP5 as a lightweight web application fulfilling the previously stated goals. It should be customizable via configuration items, e.g. for the URL of the Directory. Eventually a rate limiter should be included into the API.

No user interface is intended for this service.

The XML schema of the IAL responses is provided inside the WP5 development project at https://github.com/de4a-wp5/ial-service

3.4.3 Operations

The IAL Proxy Service is an open service that can be queried by any one knowing the URL. As it operates solely with open data, no additional security measures needed to be provided.

4 IAL Tasks

This chapter outlines the tasks for the different actors for the different components. It consists mainly of bullet points, as the exact details of the actions are not known in all details.

4.1 Tasks for WP5

This subchapter deals with the requirements for WP5.

- Directory
  - Eventually update the SMP trust store to include the TLS certificate for the Directory. This can be avoided by deploying the Directory on the same domain as most of the other Playground components.
- IAL Proxy Service
  - Implement the service
- Connector implementation
  - Replace mock IAL with real IAL calls
  - Make the URL of the IAL service customizable
  - Eventually update the Connector trust store to include the TLS certificate for the IAL Proxy Service. This can be avoided by deploying the IAL Proxy Service on the same domain as most of the other Playground components.
  - Update the API described in chapter 2.1.1 to the new logic
- General tasks
  - Document the solution for the upcoming project Deliverables

4.2 Tasks for Playground operator

This subchapter deals with the requirements for the operator of the DE4A Playground. Both components are intended to be deployed similar to the existing Playground components, to limit the effect on pilot partners.

- Directory
  - Create the configuration settings
  - Deploy the Directory once in the Playground
  - Ensure secure Internet access via TLS

[2] See https://ec.europa.eu/eurostat/web/nuts/national-structures

- IAL Proxy Service
  - Create the configuration settings
  - Deploy the IAL Proxy Service once in the Playground
  - Ensure secure Internet access via TLS

4.3 Tasks for pilot partners

This subchapter deals with the requirements for all pilot partners.

- SMP tasks
  - Configure SMP to connect to Directory and enable the automatic update of all Business Cards
  - Create and fill Business Cards for the Participant Identifiers they own
  - Eventually open firewall ports to the Directory. This can be avoided by deploying the Directory on the same domain as most of the other Playground components.
  - Deploy updated trust store with Directory TLS certificate. This can be avoided by deploying the Directory on the same domain as most of the other Playground components.
- Connector tasks
  - Deploy new Connector versions that use the new IAL
    - Should contain updated trust store with IAL Proxy Service TLS certificate. This can be avoided by deploying the IAL Proxy Service on the same domain as most of the other Playground components.
  - Eventually open firewall ports to the IAL proxy service. This can be avoided by deploying the IAL Proxy Service on the same domain as most of the other Playground components.