# dataModelTranslation

## Service Description

**Abstract**
This document provides service description for the **dataModelTranslation** service.

# Contents

**1 Overview** **3**

1.1 How This Service Is Meant to Be Used . . . 3
1.2 Important Delimitations . . . 3
1.3 Access policy . . . 3

**2 Service Operations** **4**

2.1 operation init-translation . . . 4
2.2 operation get-translation-result . . . 4
2.3 operation abort-translation . . . 4

**3 Information Model** **5**

3.1 struct DataModelTranslationInitRequest . . . 5
3.2 struct Identity . . . 5
3.3 struct Map . . . 5
3.4 struct ErrorResponse . . . 5
3.5 struct DataModelTranslationResultResponse . . . 6
3.6 Primitives . . . 6

**4 References** **7**

**5 Revision History** **8**

5.1 Amendments . . . 8
5.2 Quality Assurance . . . 8

# 1 Overview

This document describes the **dataModelTranslation** service, which provides functionality for translating between one or more data model pairs. A data model is expected to be referenced by a unique ID, and it defines both a specific format (e.g., JSON, XML) and its associated semantics. To enable other systems to consume this service, it needs to be offered through the ServiceRegistry and the handled data model IDs have to be included as part of the service metadata.

The **dataModelTranslation** service contains the following operations:

- *init-translation* accepts the data payload that need to be translated, creates and internal task for it and immediately returns a unique task ID;
- *get-translation-result* returns the actual status of a given task and the translated data payload if the task has already finished;
- *abort-translation* terminates and removes a given task;

The rest of this document is organized as follows. In Section 2, we describe the abstract message operations provided by the service. In Section 3, we end the document by presenting the data types used by the mentioned operations.

## 1.1 How This Service Is Meant to Be Used

The data model translation process occurs asynchronously. Service consumers have to first call the *init-translation* operation with the data payload to be translated. If the translation task is successfully initialized, the operation returns a unique task identifier. Using the received task ID, the *get-translation-result* operation has to be polled at a chosen frequency. Until the translation task is completed, the operation returns only the current status; once the task is finished, it returns the translated data payload or an error message if the task has been failed.

## 1.2 Important Delimitations

Data translation is only possible between data models and in the directions that are defined in the service metadata.

## 1.3 Access policy

Available for anyone within the Local Cloud with proper permissions.

# 2 Service Operations

This section describes the abstract signatures of each operations of the service. The **dataModelTranslation** service is used to initialize, check and abort a translation task. In particular, each subsection names an operation, an input type and one or two output types (unsuccessful operations can return different structure), in that order. The input type is named inside parentheses, while the output type is preceded by a colon. If the operation has two output types, they are separated by a slash. Input and output types are only denoted when accepted or returned, respectively, by the operation in question. All abstract data types named in this section are defined in Section 3.

## 2.1 operation init-translation (DataModelTranslationInitRequest) : TaskID / ErrorResponse

Operation *init-translation* returns a unique task ID when the translation task has been successfully initialized. The operation returns immediately and the translation task is executed asynchronously.

## 2.2 operation get-translation-result (TaskID) : DataModelTranslationResultResponse / ErrorResponse

Operation *get-translation-result* returns the actual status of a given translation task, the translated data payload if the task has completed successfully, or an error message if the task has failed.

## 2.3 operation abort-translation (TaskID) : Void / ErrorResponse

Operation *abort-translation* terminates and removes a given translation task.

# 3 Information Model

Here, all data objects that can be part of the **dataModelTranslation** service are listed and must be respected by the hosting system. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is used to denote a collection of named fields, each with its own data type. As a complement to the explicitly defined types in this section, there is also a list of implicit primitive types in Section 3.6, which are used to represent things like hashes and identifiers.

## 3.1 struct DataModelTranslationInitRequest

| Field | Type | Mandatory | Description |
|---|---|---|---|
| authentication | Identity | yes | The requester of the operation. |
| inputModelId | String | yes | The identifier of the data model to be translated. |
| outputModelId | String | yes | The identifier of the required result data model. |
| payload | Object | yes | The input data to be translated. |
| settings | Map | no | Any configuration data that might be required by a translation process. |

## 3.2 struct Identity

An Object which describes the identity of a system.

## 3.3 struct Map

An Object which maps String keys to primitive, Object or list values.

## 3.4 struct ErrorResponse

| Field | Type | Description |
|---|---|---|
| status | OperationStatus | Status of the operation. |
| errorMessage | String | Description of the error. |
| errorCode | Number | Numerical code of the error. |
| type | ErrorType | Type of the error. |
| origin | String | Origin of the error. |

## 3.5 struct DataModelTranslationResultResponse

| Field | Type | Description |
|---|---|---|
| status | TaskStatus | Status of the translation task. |
| result | Object | The result (translated) data payload. |
| mimeType | MimeType | The MIME type of the result data payload. |

## 3.6 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

| Type | Description |
|---|---|
| ErrorType | Any suitable type chosen by the implementor of service. |
| MimeType | Standardized identifier that defines the nature and format of a data payload. |
| Number | Decimal number. |
| Object | Set of primitives and possible further objects. |
| OperationStatus | Logical, textual or numerical value that indicates whether an operation is a success or a failure. Multiple values can be used for success and error cases to give additional information about the nature of the result. |
| String | A chain of characters. |
| TaskID | A unique string identifier that belongs to a translation task. |
| TaskStatus | Logical, textual or numerical value that indicates whether a translation task is pending, in progress, finished or failed. |

# 4 References

# 5 Revision History

## 5.1 Amendments

| No. | Date | Version | Subject of Amendments | Author |
|---|---|---|---|---|
| 1 | YYYY-MM-DD | 5.1.0 | | Xxx Yyy |

## 5.2 Quality Assurance

| No. | Date | Version | Approved by |
|---|---|---|---|
| 1 | YYYY-MM-DD | 5.1.0 | |