STSOrgSync

Solution Description

**Version**: 1.0.0

**Date**: 25.11.2015

**Author**: BSG

Content

[1 Introduction 3](#_Toc465665064)

[1.1 Background 3](#_Toc465665065)

[1.2 Purpose of STSOrgSync 3](#_Toc465665066)

[2 Functionality 4](#_Toc465665067)

[3 Design 5](#_Toc465665068)

[3.1 ServiceLayer 5](#_Toc465665069)

[3.2 SchedulingLayer 5](#_Toc465665070)

[3.3 BusinessLayer 6](#_Toc465665071)

[3.4 IntegrationLayer 6](#_Toc465665072)

[4 Usage 6](#_Toc465665073)

[4.1 Using an existing integration 6](#_Toc465665074)

[4.2 Choosing an integration pattern for a custom integration 7](#_Toc465665075)

[4.2.1 Service integration 7](#_Toc465665076)

[4.2.2 SQL integration 7](#_Toc465665077)

[4.2.3 SDK integration 7](#_Toc465665078)

# Introduction

This is the entry document, which should be read before any of the other documents. It contains all the general information needed to understand the STSOrgSync solution, and be able to read and use the other documents.

## Background

KOMBIT is delivering 8 infrastructure components (called Støttesystemerne), one of which is called Organisation. The purpose of Organisation is to be a uniform integration point for it-systems, meaning

* All municipalities must register and maintain their organizational data inside Organisation.
* It-systems that need to read and use municipality organizational data will integrate against Organisation, and not require the municipalities to perform custom integration or configuration for that specific it-system.

The first it-systems that need to read from Organisation are

* KY (Kommunernes Ydelsessystem)
* SAPA (Sagsoverblik/Partskontakt)
* KSD (Kommunernes Sygedagpengesystem)
* DHUV (Digitalisering på Handicap- og Udsatte Voksne-området)



This integration pattern has several long-term benefits, including

* All it-systems will use the same organization data, increasing quality of data
* No custom integrations required for each municipality, reducing cost of acquiring the it-system

Over time more and more it-systems will integrate with Organisation, increasing the value of the data in Organisation.

## Purpose of STSOrgSync

The primary purpose of the STSOrgSync solution, is to allow municipalities to easily register and maintain data in Organisation, based on existing local data.

The following goals have been set for STSOrgSync

* It must handle all data required by Organisation, so no manual maintenance of data in Organisation is required.
* It must not require a specific local system for storing data, but should allow easy integration with any local system.
* It must not require that local data is modelled in a specific way, but should be able to transform local data into the OIO model used by Organisation.
* It must not require that local data contains historic information, but should instead build and maintain historical data inside Organisation
* It must conform to the registration patterns outlined by KOMBIT

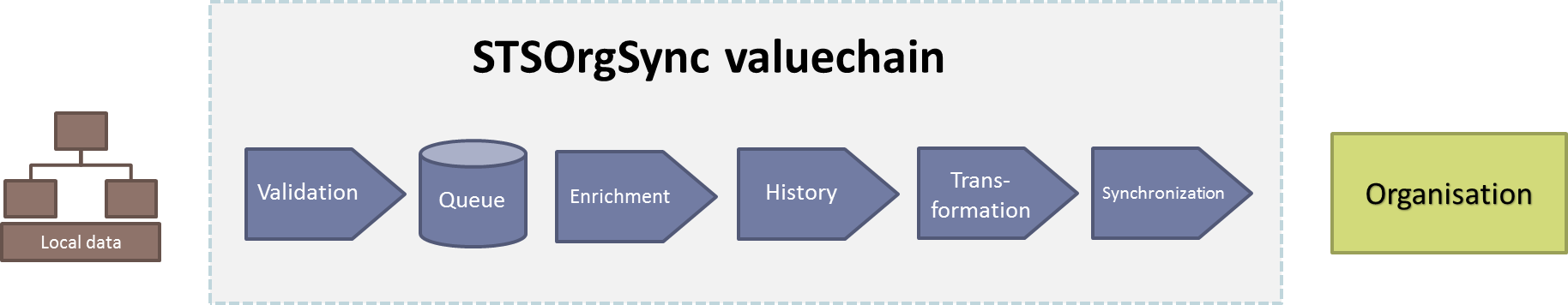
# Functionality

The solution supports the enrichment, transformation and synchronization of 3 different local datatypes

* Users
* Organizational Units
* It-systems

Whenever one of these objects change state locally, the STSOrgSync should be informed of the objects new state, after which it will ensure that the data is properly synchronized with Organisation. This process is outlined in the flowchart below

*Note that the flowchart shows the full value-chain. Depending on the chosen integration pattern (see chapter 4.2 for details) some of these steps might be bypassed.*



Every event on a local object is processed in the same way

1. The object is validated, to ensure that the local data conforms to the minimum requirements for storing data inside Organisation. If not, the object is rejected.
2. The object is put into a queue for later processing. This ensures that any network or system-related issues does not prevent the object from being synchronized.
   1. Note that the queue functionality is bypassed when the SDK integration pattern(see chapter 4.2 for details) is used in STSOrgSync. The queue functionality is used by the Service and SQL integrations offered by STSOrgSync.
3. If the object does not already exist in Organisation, the following happens
   1. the object is enriched with various keys required by the OIO data model
   2. the object is transformed to the OIO data model
   3. the object is imported into Organisation
4. If the object already exists in Organisation, the following happens
   1. The local data is compared with the existing object in Organisation, and the list of changed attributes is extracted
   2. For each modified attribute, the object in Organisation is updated while ensuring that historical data is stored on the object correctly
   3. The updated object is registered in Organisation

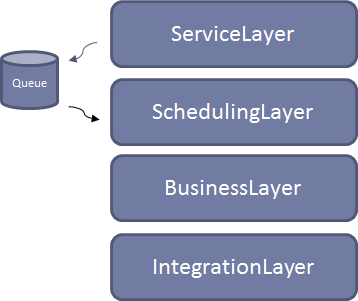
Note that the OIO Organisation data model consists of many different object types, which means that a single local object (e.g. a User) will be transformed to multiple objects in Organisation.

For instance, a user that has an email address and a phone number, is converted into 5 objects in the OIO data model (a User object, a Person object, 2 Address objects and finally an OrganisationFunktion object which is used to map the user to the Organizational Unit that the user works in).

The STSOrgSync transformation logic handles this mapping of 1-to-many objects.

# Design

The STSOrgSync consists of 4 separate modules, each module is responsible for a specific set of tasks. The modules are layered, so when using the “top” module, it automatically encapsulates the functionality found in the other modules.



## ServiceLayer

The ServiceLayer is a web service module, which exposes 3 JSON/REST service endpoints. It accepts local objects as input, and stored the objects in the queue if they pass initial validation.

The ServiceLayer is deployable as a stand-alone application, and can be installed on a Windows Server 2012 R2 as a Windows Service. Once installed, it can be integrated with by the local system(s) that manages the municipalities organizational data.

It requires access to a Microsoft SQL Server, which is uses as the queueing mechanism, for temporarily storing data if the Organisation system at KOMBIT cannot be reached.

The ServiceLayer depends on the other three modules.

## SchedulingLayer

The SchedulingLayer is an automated task, that at regular (configurable) intervals pulls data from the queue, and attempt to synchronize them with Organisation.

In case of errors, the type of error decides if the data is put back onto the queue, or if the data is dropped.

It is possible to bypass the ServiceLayer and use the SchedulingLayer directly. This is done by writing events directly to the queue, performing SQL Insert operations rather than calling the JSON/REST service. Input validation is ensured by schema restrictions.

If this integration is chosen, simply deploy the same Windows Service that contains the ServiceLayer, and write directly to the queue instead of calling the Windows Service.

The SQL Schema used by the SchedulingLayer is a public interface, and will be maintained as such – so any future changes to STSOrgSync will take this into consideration.

## BusinessLayer

The BusinessLayer is responsible for most of the business logic in the STSOrgSync solution, including enrichment, transformation and maintenance of historic information.

The BusinessLayer is used by the SchedulingLayer when it synchronizes data with Organisation, but it is also a fully functional SDK written in .NET, allowing a tight integration of the STSOrgSync functionality into existing applications, without having to use the Service or SQL integrations.

The BusinessLayer performs the same input validation as the ServiceLayer, but the programmer using the SDK will have to do error handling in case the Organisation system is down. The SDK throws specific exceptions to indicate whether the error is temporary in nature (and the programmer should retry later) or if the error is permanent, and the data needs to be fixed before it can be synchronized.

## IntegrationLayer

The IntegrationLayer performs one of the simplest operations in the STSOrgSync solution, but also does most of the work. It is responsible for handling security (certificates, tokens, etc) and for reading and writing data to Organisation.

The IntegrationLayer is not intended as a module that is used directly, and it is specifically tailored to support the needs of the BusinessLayer. While it is possible to use the functionality in the IntegrationLayer directly, it is not recommended.

# Usage

Due to the flexibility of the STSOrgSync solution, using it is not as simple as just installing the software. The solution needs to be integrated with one (or more) it-systems running locally, from where it can read the data that it needs to register in Organisation.

## Using an existing integration

While there are 98 municipalities in Denmark, there are not 98 different systems in which municipalities store their organizational data.

Quite a few uses Active Directory, while others use more feature rich organization systems like SOFD, APOS, BSK, Frille, Person Stamdatabasen etc.

There exist integrations for several of these systems already, and using these can be as easy as installing and configuring the software, and then everything just works without further customization.

The STSOrgSync project will keep a list of available integrations, which can either be used as-is, modified as needed or simply serve as an inspiration for a custom integration.

## Choosing an integration pattern for a custom integration

If no useful existing integration exist, then implementing a custom integration is an easy project to get started with, and also a very small project.

As STSOrgSync offers 3 different ways to integrate with the solution, one such integration pattern must be chosen before implementation can begin.

### Service integration

This integration pattern has the advantage that STSOrgSync is installed as a stand-alone application, and that it exposes very simple REST/JSON services for maintaining data in Organisation.

Integrating against STSOrgSync can be very simple, if the system storing the local data is capable of calling a REST/JSON service when data is changed in the local system.

### SQL integration

This integration pattern has the same advantages as the Service integration, but offers an SQL interface rather than a REST/JSON interface.

If the local data is stored in a SQL database, it might be very easy to setup some SQL job (ETL or otherwise) that copies the relevant local data into the SQL queue used by STSOrgSync at regular intervals.

### SDK integration

Finally, STSOrgSync allows for integration on the code-level. If neither the Service or SQL integrations are a good match, then using the BusinessLayer SDK directly is also an easy approach to synchronizing data.

The BusinessLayer SDK offers the exact same API as the SQL and Service Layer, and only minimal additional error/retry handling is required as compared to the other integrations.