Insights into OCDS implementation

Introduction

The aim of this report is to provide insights into the technical choices made in OCDS implementations.

Some publishers add an OCDS export to existing procurement systems, while others implement OCDS as part of a new system. The focus of this report is on the implementation of OCDS, rather than on the implementation of procurement systems in general.

OCDS implementations can differ in a number of ways:

- **Publication formats:** JSON, CSV or spreadsheet.
- **Access methods:** Static files or API.
- **Change history:** Full releases and records, or the latest information only.
- **Architecture:** Publishing data directly or implementing a middleware.
- **Technology:** Choice of databases, frameworks and programming languages.
- **Source systems:** One or many systems, volume of data.
- **Coverage:** Which OCDS sections and fields are published.

In this report we document five different OCDS implementations. Each implementation has some features of particular interest:

- **Paraguay** uses new and existing open-source tools to publish a full change history and bulk download data.
- **Zambia** added a new OCDS publication module to a commercial e-procurement solution.
- **Colombia** publishes data on millions of contracting processes, joining 3 different systems.
- **Moldova** created a brand new multi-platform procurement system based on OCDS.
- **Argentina Vialidad** reuses an existing OCDS visualization tool and features a graphical ETL interface.

Case studies

Zambia

Zambia started publishing OCDS data in 2017 following the launch of a new e-GP system provided by European Dynamics.

To publish OCDS data, European Dynamics added a module to their existing ePPS product. The reasons for choosing to add a new module, instead of extending existing modules, were:

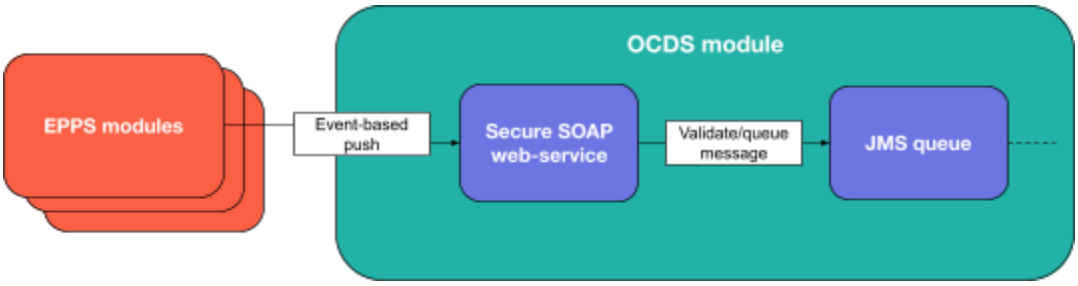
- Easing integration of new ePPS modules with the OCDS publication module.
- Avoiding performance degradation of the main ePPS system.

Extracting data

Zambia's OCDS publication is event-based. Specific events in the core ePPS system trigger a call to the OCDS publication module.

The OCDS publication module pushes messages to an asynchronous queue for processing. Using an asynchronous queue frees up resources from the core system.

Communication between the modules is via a SOAP web-service.



Transforming and loading data

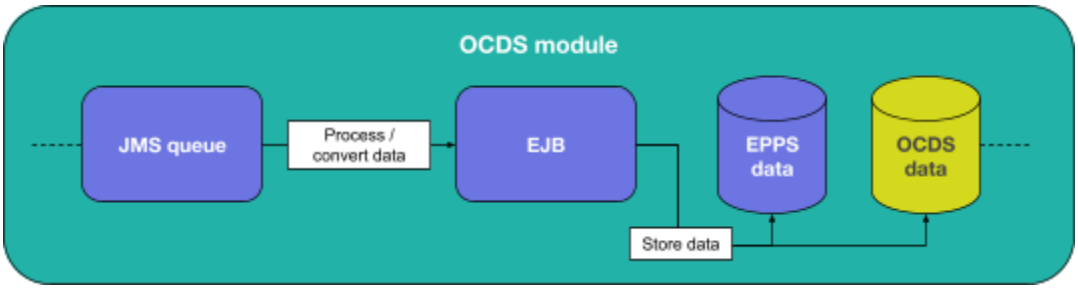
The OCDS publication module maintains two datastores, separate from the main ePPS system:

- The first store handles data exactly as communicated by the core ePPS system.
- The second store handles data in OCDS format

The main reason for choosing to keep the 'raw' data was to accommodate changes in future versions of OCDS. This approach supports the definition of new conversion processes for existing data.

The OCDS publication module transforms data to OCDS format overnight using Enterprise JavaBeans. This keeps the extra load to periods of low demand on the ePPS system.

Data storage is handled by MongoDB thanks to its support for NoSQL data.



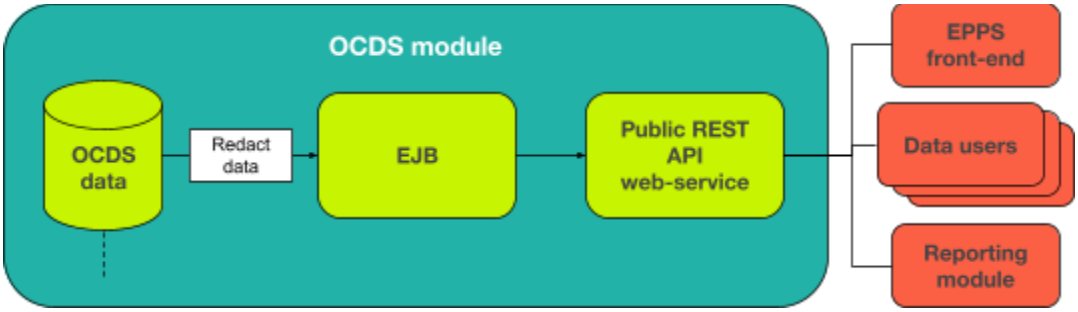
Publishing data

Zambia publishes monthly bulk record package downloads. Each record contains:

- a releases list, containing embedded copies of all releases for the contracting process
- a compiled release, with the latest version of the data about the contracting process
- a versioned release, containing a change history of each field

Bulk record packages are published via the ePPS front-end and a REST API. Individual compiled and versioned releases are also published, via the ePPS front-end.

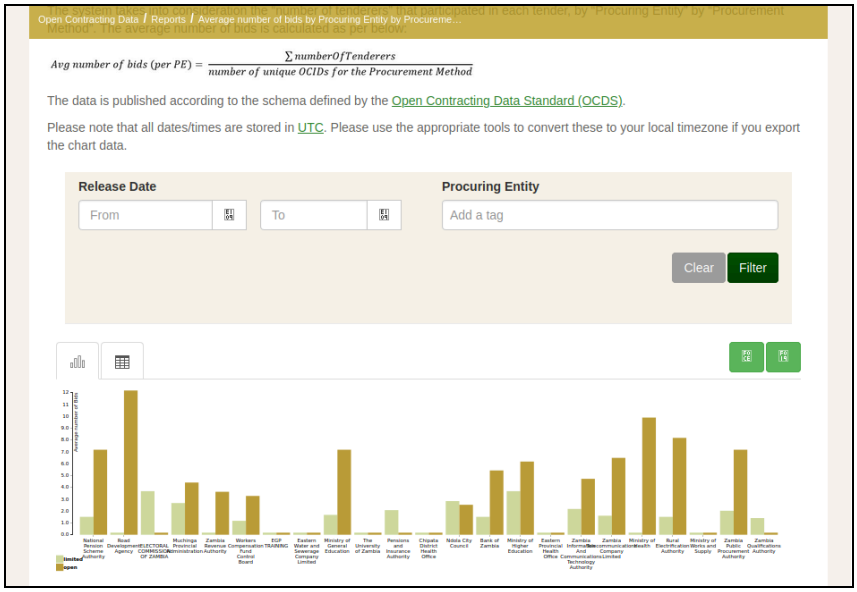
The publication module includes an extra layer on top of the OCDS datastore. This layer allows redaction of sensitive information to ensure compliance with national legislation.



Zambia publishes data in the planning, tender, awards and contracts sections of OCDS. Data is published in OCDS 1.0 format and does not use extensions.

Using data

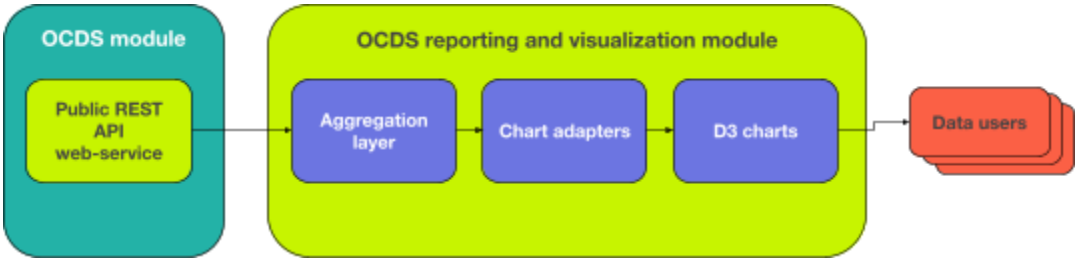
Zambia's implementation includes a visualization and reporting module. This module uses the OCDS datastore as its data source.



An OCDS aggregation layer summarizes data for visualization and reporting. MongoDB's Aggregation Pipeline and Spring Data underpin this layer. This approach reuses as much of the existing implementation as possible.

D3 handles visualization of the aggregated data. The benefits of using D3 are:

- Support for JSON-like structures reduces the data alternation required.
- Pre-cooked and highlight customizable charts
- Support for geo-location



Paraguay

The National Directorate for Public Procurement in Paraguay (DNCP in Spanish) was one of the first OCDS publishers. Back in 2016 they published OCDS 1.0 releases and records for approximately 15K processes per year. In 2019 DNCP decided to update their implementation to incorporate the changes in OCDS 1.1 and to make it easier to maintain by themselves.

The new implementation was supported by the Open Contracting Partnership and includes the publication of releases and records via an API. It also includes bulk downloads in CSV format and visualizations using aggregated OCDS data.

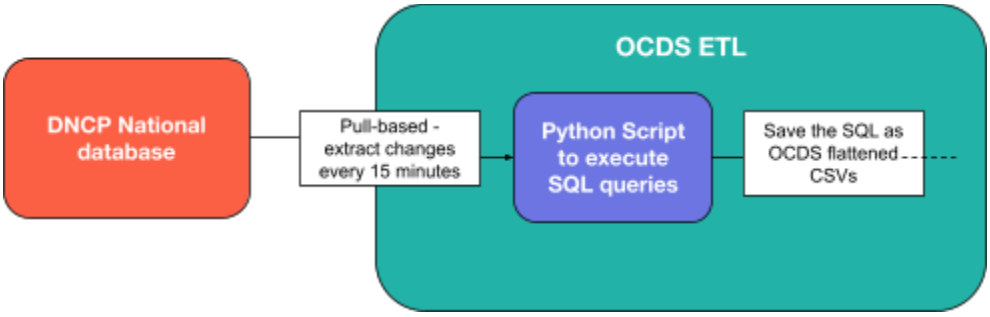
DNCP expanded the scope of the data they publish to include the following, all in a single dataset:

- All procurement methods, from direct to open.
- Multi-stage procedures, such as framework agreements and pre qualifications.
- Contracting processes from all the public entities in Paraguay.

Extracting data

Paraguay's OCDS publication is pull-based. Every 15 minutes a Python script runs a set of SQL queries to search for new changes and extract data. DNCP wanted to publish data as close to real-time as possible, but due to the processing time for the script they decided to run it every 15 minutes.

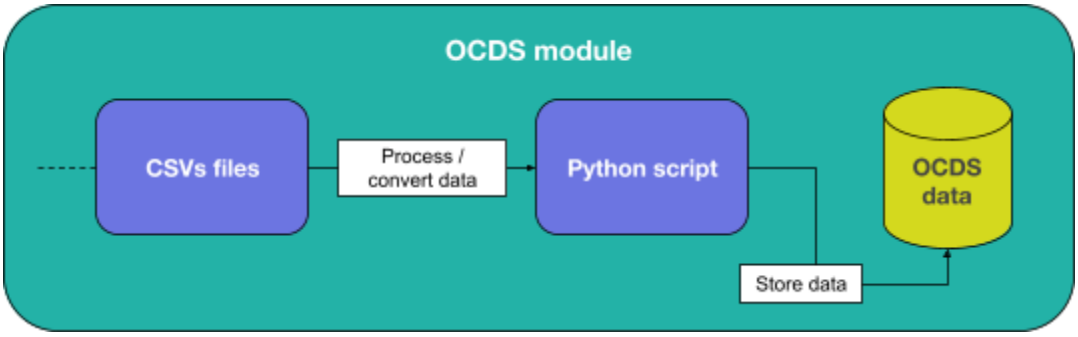
The script saves the results as CSV files in flattened OCDS format. DNCP decided to implement a SQL-based approach as they are used to working with SQL queries in their day-to-day activities.



Transforming and loading data

A Python script converts the CSVs into OCDS JSON format, using the Flatten Tool Python library. Then, the JSON files are inserted into an ElasticSearch database with two different indexes.

First, the script inserts the JSON files into the 'releases' index. Then, it queries the ElasticSearch database to find the releases for each contracting process. Then, the script generates a record for each contracting process and inserts them into the 'records' index. The script uses OCDS Kit's merge method to create records from releases.



Publishing data

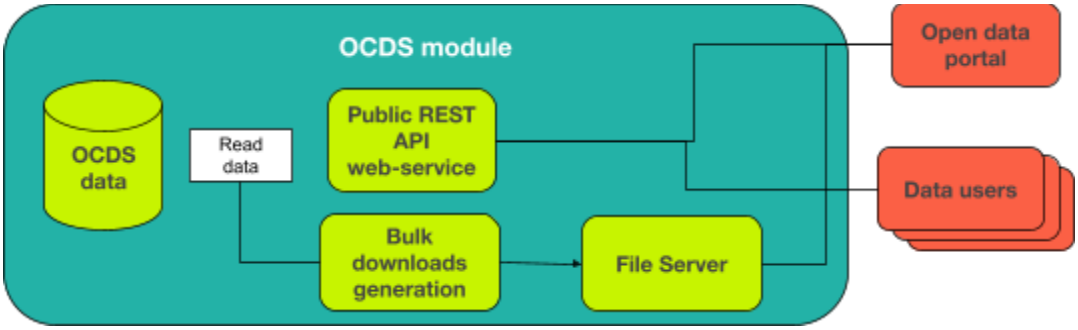
The DNCP publishes the data through an API and through bulk downloads in CSV format.

The API has three OCDS API endpoints:

- release by id
- releases by ocid
- record by ocid

The API has other services such as searching processes by any OCDS field. The API was developed using the Flask Python framework, consuming the data stored in the ElasticSearch database. The tool is available as open source.

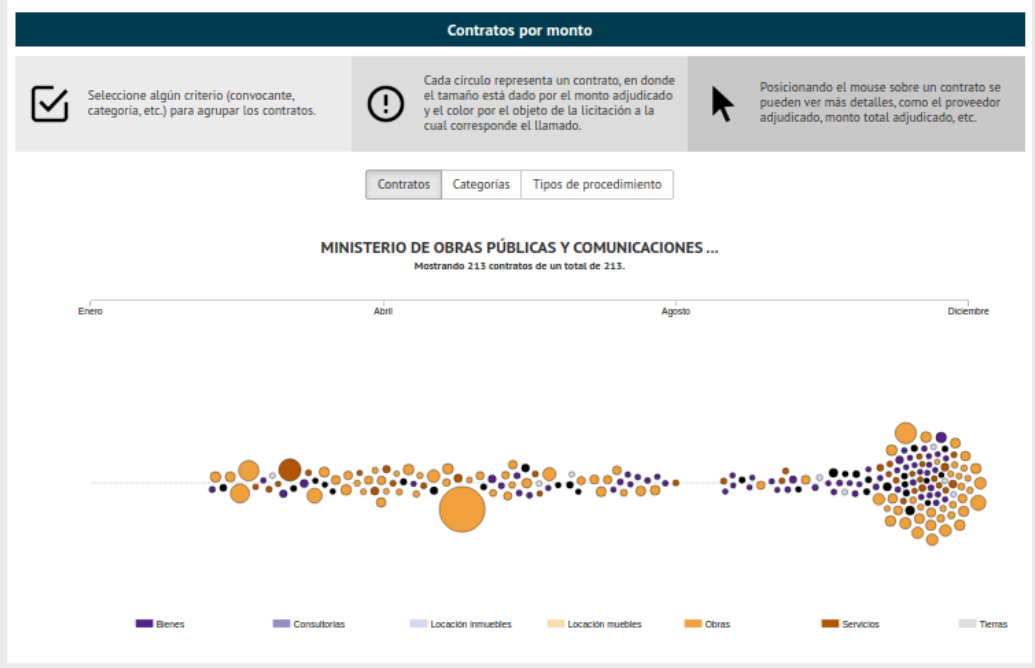
The bulk download consists of flattened version of all the OCDS records. These CSVs are generated through a Python script. The script queries the ElasticSearch database and uses Flatten Tool to convert the results from JSON to CSV.



Paraguay publishes data in the planning, tender, awards, contracts and implementation sections of OCDS. Data is published in OCDS 1.1 format and uses a set of local, community and core extensions. DCNP includes, through extensions, data related to auctions, complaints, budget breakdowns, pre-qualifications, framework agreements, planned items, related investment projects, and more.

Using data

DNCP uses the API to power its open source visualizations.



The visualizations use aggregated and summarized OCDS data to show:

- The contract date distribution per month and year by procuring entity
- The contract amount distribution by procuring entity and procurement method
- The different stages of a given contracting process
- A calendar of the upcoming tenders with their opening dates.

The Inter-American Development Bank also uses the DNCP OCDS data to join with data about investment projects and its contracts.

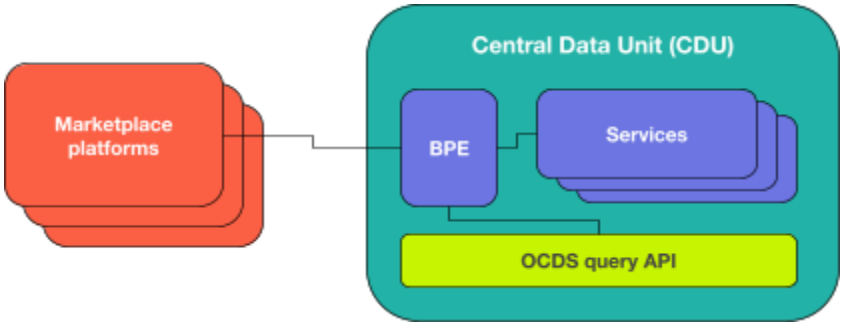
Moldova

Moldova committed to increase procurement transparency in its 2016-18 OGP Action Plan. the Ministry of Finance pledged to develop a new e-procurement system based on OCDS, inspired by Ukraine's Prozorro system. The European Bank for Reconstruction and Development supported the development of the system.

The new system, MTender, uses a multi-platform architecture. Buyers and suppliers conduct processes using commercial marketplace platforms. A Central Data Unit (CDU) connects the marketplaces and provides access to e-government services. This multi-platform architecture aims to:

- promote stability
- increase competition
- reduce political influence
- facilitate transparent cross-border e-procurement

The developers of MTender used OCDS to inform the design of the internal databases and transactional functions. The system uses OCDS-based data models for internal transactions and for communication between marketplaces and the CDU. A Business Process Engine (BPE) coordinates operations between the marketplaces and the CDU. The CDU includes an OCDS query API component for publishing data directly from the system.



Moldova uses the new system to manage competitive procedures only and so far it contains c. 60,000 contracting processes.

Extracting data

Moldova's OCDS publication is event-based. After processing each operation, the BPE pushes data to the OCDS query API component in real-time.

All the business processes in the system are also event-driven and controlled by Camunda BPM. This is an open-source technology chosen as the main process management framework since it supports the use of BPM 2.0, an executable notation for automated business logic processing which allows flexibility “by design” (modeling instead of coding). Kafka is used for messaging.

Transforming and loading data

Currently, data from the BPE is not transformed, since it is already in an OCDS-based format.

The OCDS query API maintains its own distributed transactional database of all the data-outcomes. These are stored immutably, separate from the other CDU components.

Apache Hadoop handles data storage, using an Apache Cassandra database.

Publishing data

MTender currently publishes 3 OCDS API endpoints:

- Budgets - information on the budget lines and/or related projects through which contracting processes are funded
- Plans - announcements of the intention to procure some goods, works or services
- Tenders - details of the procurement procedures initiated by procuring entities

Each endpoint publishes OCDS records in real-time in JSON format, using an out of the box Hadoop API. The reason for choosing Hadoop was because of its ability to handle high loads on the OCDS query API.

Data structure

Because the API is designed to meet the needs of the marketplace platforms and to facilitate cross-border transactions, the Mtender public endpoint diverged from OCDS 1.1. Data about each contracting process was split across the endpoints in order to separate data related to multiple contracting processes from data related to a specific process only. Each endpoint assigned a different OCID to each part of data.

This normalization allows changes resulting from one contracting process to be reflected in other contracting processes, for example where the conclusion of a contract leads to a reduction in a budget line, this is reflected in the budget available for subsequent contracting processes.

Because OCDS 1.1 requires all the data about a single contracting process to be published under one OCID, MTender developers added an extra OCDS 1.1 conformant endpoint -

http://public.eprocurement.systems/ocds/tenders. The new endpoint publishes all the data about a single contracting process under one OCID, which is essential for data users interested in monitoring and analysis.

Data in the new endpoint is transformed using materialized views. This approach means that the existing OCDS query API components are reused to publish OCDS conformant data, without the need for a new middleware or database.

To get full access to OCDS 1.1 compliant data, any user can retrieve the list of existing OCIDs from MTender public point: https://public.mtender.gov.md/tenders/ (you can iterate the list using the 'offset' parameter provided in the data) and then collect OCDS 1.1 data from OCDS publication point http://public.eprocurement.systems/ocds/tenders/{ocid}. You can also use Kingfisher Collect, a tool developed by the Open Contracting Partnership, to get MTender data.

Coverage

So far MTender publishes data in the planning, tender, awards and contracts sections of OCDS and uses many extensions, including bid statistics and details, enquiries, lots and requirements. This coverage is planned to extend further.

Besides contracting data, MTender also uses OCDS-based models to publish data on local review proceedings and interactions with the State Treasury.

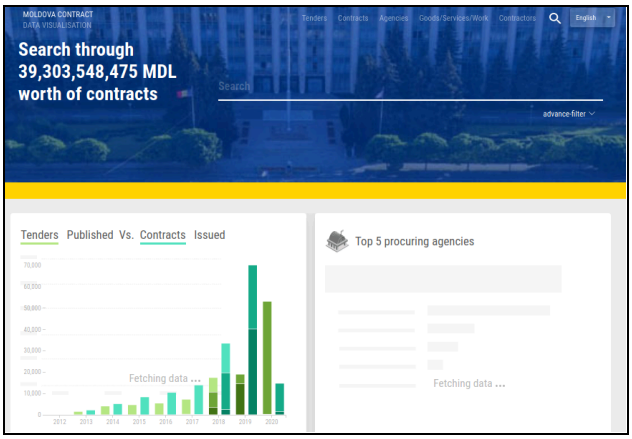
Using data

Marketplace platforms use the OCDS query API to retrieve updated data from the CDU after each operation. This approach makes it easier for platforms to keep their data in sync with the CDU, reducing barriers to entry for new marketplaces.

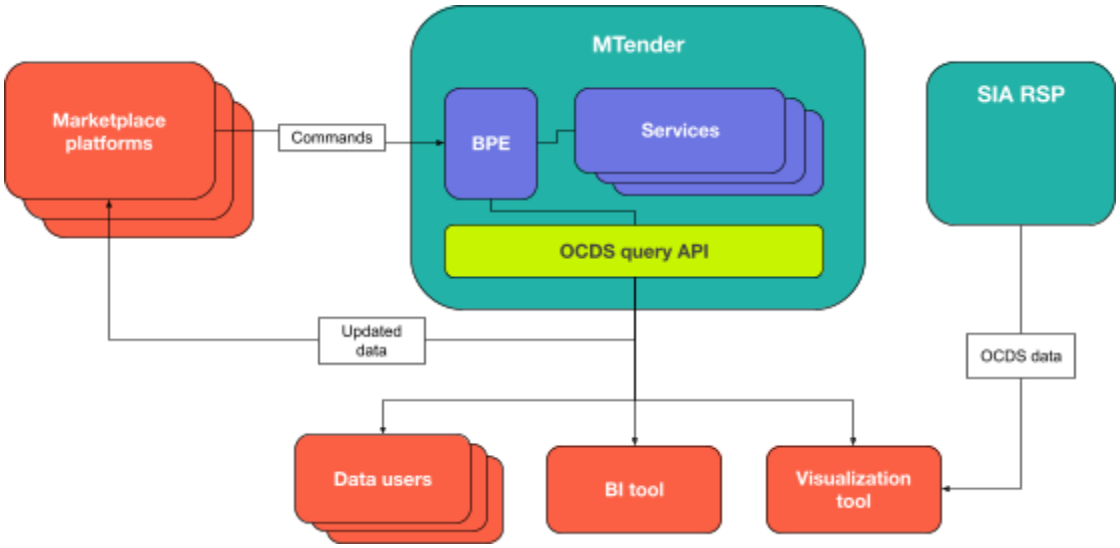
Putting the use of open data at the center of MTender creates a feedback loop between publisher and user, leading to higher quality data.

Moldova has also developed a real-time business intelligence tool based on the OCDS query API. However, access to the tool is restricted to government agencies.

Alongside MTender, Moldova uses the SIA RSAP system, which also publishes OCDS data. Data from the two systems is combined in a visualization tool to give a complete view of public contracting in Moldova.



The visualization tool is based on an [open-source tool] originally developed to display the OCDS data from SIA RSAP. This is a great example of reusing an open-source OCDS tool with data from a different system.



Future plans

MTender's developer plans to add an extra OCDS 1.1 conformant real-time publication. The new endpoint will enable the use of [open source analytical OCDS tools] in Moldova.

A number of functional add-ons to MTender are already under development. Among them are support for framework agreements, electronic catalogs, utilities-specific procurement method, document workflow management and contract implementation.

EBRD believes that the implementation of multi-platform solutions, such as MTender, which use OCDS-based models on a transactional level can ease cross-border

procurement. For example, by enabling suppliers to bid for cross-border opportunities using a local marketplace.

Colombia

Colombia Compra Eficiente (CCE) started publishing OCDS 1.0 data back in 2015. In 2019, in collaboration with the Open Contracting Partnership, they updated their publication OCDS version 1.1. The data includes contracting processes from 2011 to now, covering around 7 million processes.

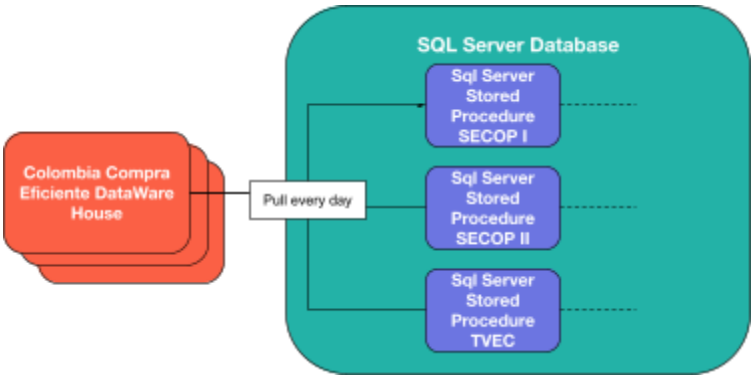
Colombia Compra Eficiente has 3 different source systems, so they decided to use OCDS to publish data from all 3 systems in a common format. This makes it easier for users to analyse data from across the different systems.

Because of the large amount of data and the number of source systems to map, they had to choose their implementation architecture with care.

Extracting data

CCE implemented a pull-based data extraction using a SQL Server Stored Procedure for each database. The procedure is executed daily to extract all the data from each system.

CCE reused their existing SQL Server 2016 Data Warehouse to extract and transform the data. This makes the process faster and easier to maintain.



Transforming and loading data

One of the biggest challenges for Colombia was the amount of data that they handle. Although OCDS encourages the publication and storage of a change history, this wasn't possible for Colombia. They instead decided to store only the latest version of the data about each contracting process; each time a new release is generated, the previous one is overwritten.

To transform the data, CCE uses SQL Server Integration Services. To load the data, they first used SQL Server Integration Services too, but then changed to .Net Core+. This made it

possible to load only new or changed data from each transformation, rather than all the data every time.

The OCDS releases are loaded into a MongoDB database, which is then used by the API.

Publishing data

Colombia publishes their data through an API and with JSON bulk downloads. The API is a Java application connected to MongoDB and documented with Swagger.

The API has different OCDS endpoints, including:

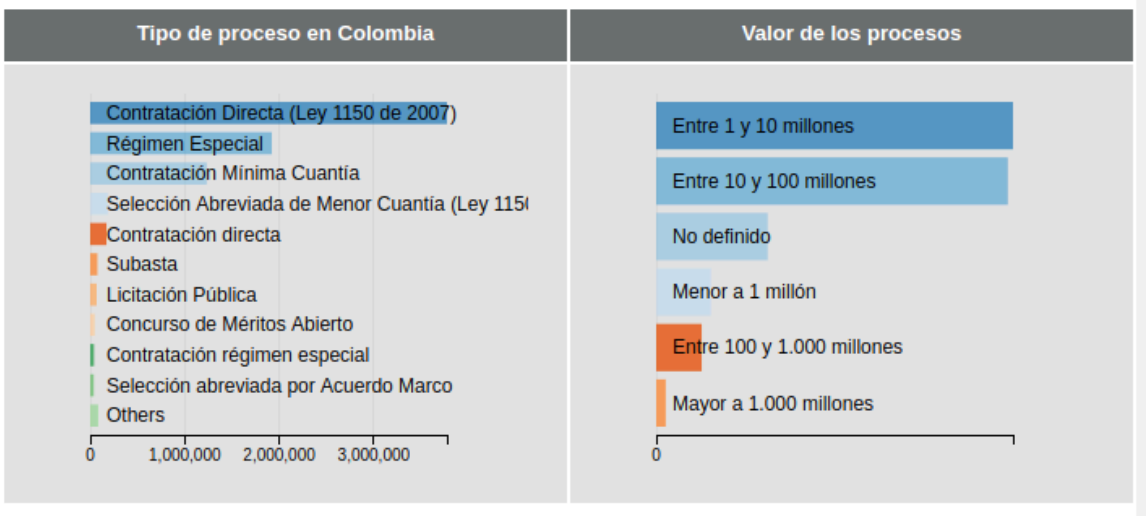
- releases: paginated list of all OCDS releases
- releases/filters: paginated list of all OCDS releases filtered by any OCDS field
- releases/year: paginated list of all OCDS releases filtered by year
- releases/date: paginated list of all OCDS releases filtered by start and end date
- releases by ocid
- releases by id

The bulk downloads are in json lines format and are separated by system, year and month. Currently the format used is not OCDS conformant, as it is not a release or record package.

Colombia publishes data in the planning, tender, award and contract sections of OCDS, and uses a number of local, community and core extensions, mainy to publish detailed information about the parties.

Using data

Colombia Compra uses its API to visualize some aggregated data about their processes.



Argentina - National Roadwork Directorate (DNV)

The National Road Work Directorate is the entity responsible for road work-related infrastructure projects and contracts in Argentina.

Part of the 2019 Argentina National Anti Corruption Plan was to:

> *"[develop] a public, open and interactive Map of Road Public Works updated periodically where the citizen can view all the works in progress that exist on each route in the country, in order to have ONE (1) homogeneous database on the works in progress, reinforcing citizen control and public accountability"*

With that commitment in mind, in early 2019, Development Gateway (DG) began collaborating with DNV to publish its contracting data using the Open Contracting Data Standard.

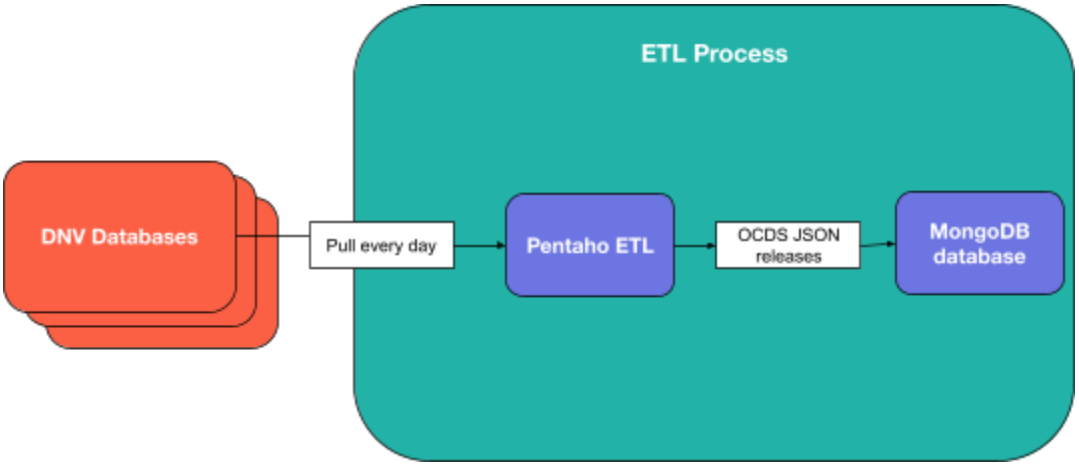
DNV publishes OCDS data via an API and bulk JSON download. The data covers from 2016 to now and includes planning, tender, award and contract data for 158 projects. DNV also built a public portal with visualizations, dashboards and a map.

Extracting and transforming data

One of the main challenges that DG faced was extracting data. The data came from three different systems and sometimes lacked common identifiers to join up data from the same contracting processes. DNV plans to update its databases in the future, so the chosen solution needed to be flexible. With all that in mind, DG and DNV decided to use Pentaho Data Integration (Kettle PDI) ETL tool.

The extraction part uses the Java Database Connectivity (JDBC) feature of Pentaho to connect and extract data from different databases. A logic node compares and joins data with matching IDs. Then, a javascript script node transforms the extracted data to JSON. Finally a REST node loads the OCDS JSON files to a MongoDB database.

The ETL process runs every night and takes only 5 minutes to be completed. The process drops the entire database and regenerates it every time, as DNV databases don't have a way to determine if a process changed or not. For that reason, they only store the last version of the process and not the changes history. As they only have 158 projects, it isn't too slow to drop and regenerate the database each time.



Loading data

The data is loaded into a MongoDB database by the Pentaho ETL. DG chose MongoDB as it is a Nosql database and the DG existing tools use this database as source.

Publishing data

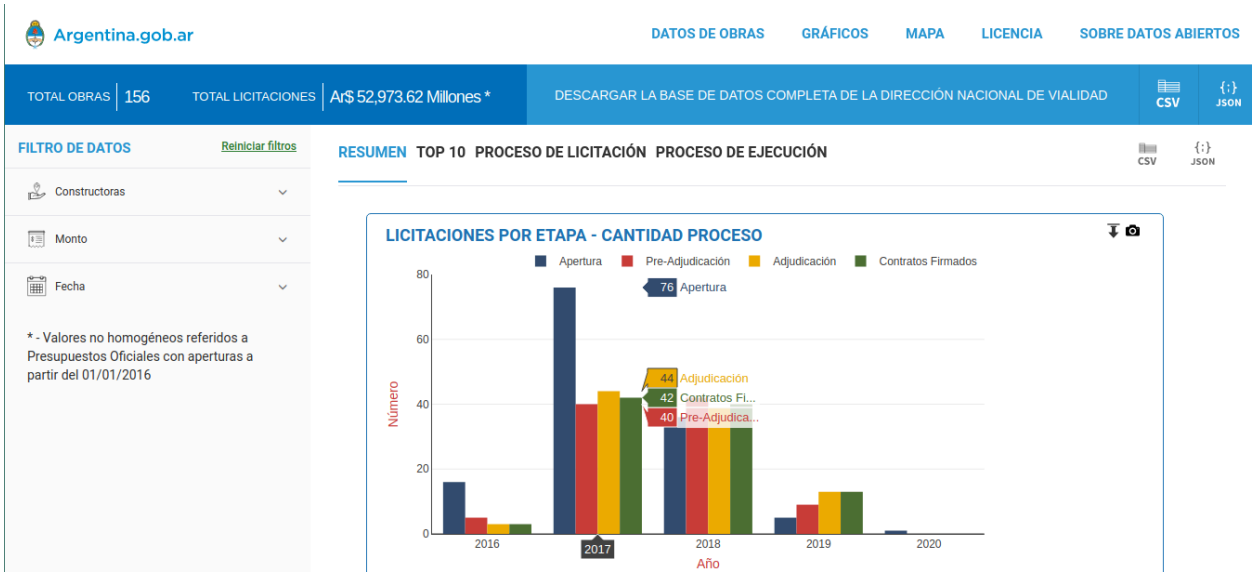
DNV publishes their data through an API and CSV and JSON bulk downloads. The API has an endpoint that returns a release package with all the existing processes.

To publish the data they used the existing DG tool, OC-Explorer. The tool consists of a Java SpringBoot application that is already used in other countries, including Kenya and Vietnam.

DNV publishes data in the planning, tender, award, contract and implementation sections of OCDS and uses a number of extensions, such as the core bid statistics and details extension and developed some local ones: as the value of the Argentine peso has drastically changed over the past years due to inflation, the DNV developed an OCDS extension that allows users to visualize updated values for planned budget and contract amounts.

Using data

The OC-Explorer tool also includes a series of visualizations and dashboards that were used to create an open data portal.



The open data portal includes statistics visualizations, a georeferenced map of projects, the projects lists and others.

Conclusions

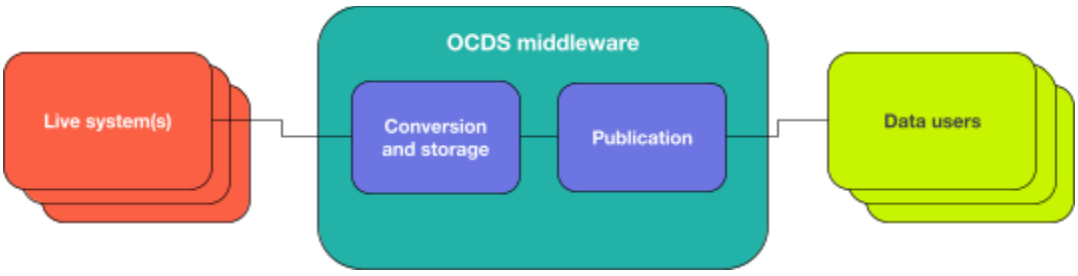
The case studies in this report show that there are many routes to publishing OCDS data. There is no one-size-fits-all solution and the best approach will depend on the context and technical constraints.

Each implementation covered in the case studies has its own strengths and weaknesses. In this section of the report we draw comparisons between the different approaches and highlight the impacts on data users.

For a full comparison of the approaches, see Appendix 1: Comparison table.

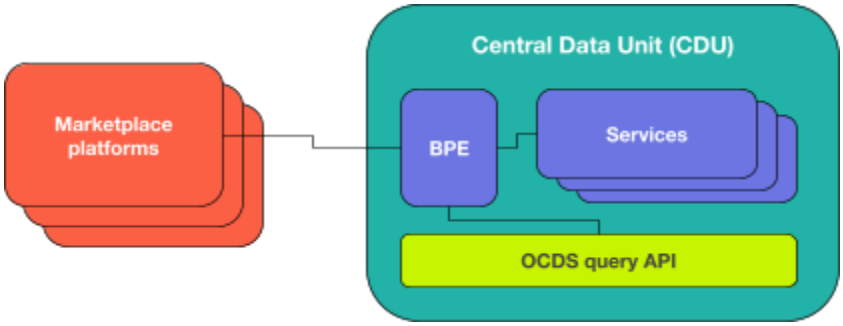
Architecture

With the exception of Moldova, all implementations discussed in this report are variations on the same high-level architecture. Data is extracted from live systems and stored in OCDS format for publication:



There are differences in how each implementation extracts, transforms, loads, and publishes data.

In Moldova's case, the OCDS query API provides an interface to the core MTender system.



Extracting data

Approaches to extracting data fall into two categories:

- In Zambia and Moldova's implementations, events in the system trigger a push of data for publication
- In Paraguay, Colombia and Argentina, data is periodically pulled from source systems.

In pull-based implementations there is a risk of data loss. This can happen when the source data changes more than once between data extractions. The frequency of extraction can mitigate this risk and varies between implementations. Paraguay extracts data every 15 minutes, whilst Colombia and Argentina extract data daily.

The number and size of source systems can affect how frequently data can be extracted. In Colombia's case, data is extracted from 3 different systems covering millions of contracting processes.

Event-based implementations reduce the risk of data loss, as each change triggers the publication of data.

Real-time access to data can also be easier to achieve in event-based systems, as seen in Moldova where data is published after each operation. Timeliness can still depend on later stages though, as in Zambia where data is transformed overnight. Transformation isn't required for Moldova's implementation since it uses OCDS-like data models for internal communication.

Pull-based publication can be simpler, since it only requires read-access to the source system database. Event-based publication requires more integration between the source system and middleware and can be harder to implement for multiple source systems.

Transforming and loading data

All publishers store data in OCDS format before publication, rather than transforming it on-demand.

In Zambia, the 'raw' data sent by the source system is also stored. The advantage of this approach is that the new conversion processes can be defined, for example to conform to future versions of OCDS. The downside is that more storage capacity is required.

Except for Moldova, where data is already in OCDS-like format, data needs to be transformed before storage. The approaches and technologies used for transformation differ between publishers.

Paraguay's approach is particularly interesting since it reuses existing OCDS tools. The tools convert OCDS releases from CSV to JSON format and merge the releases to create an OCDS record.

All publishers use NoSQL databases for loading data, with MongoDB being most common.

Publishing data

Access methods

All publishers provide a JSON API and most provide JSON bulk files too. Providing bulk files is important because it makes it easy for users to get all the data, which is a common need.

Paraguay and Colombia's APIs support searching by any OCDS field. Argentina's API supports filtering on certain parameters, such as statuses and dates. Providing search and filter functions in APIs or via an interface means that users can download only the data that they need.

Publication formats

Paraguay, Colombia and Argentina also provide CSV bulk files. Publishing data in multiple formats means that as many people as possible can use it without first having to transform it.

In particular, providing CSV format data supports users who don't know how to query JSON data or how to use command-line tools to transform it. This makes the data accessible to those used to working with SQL databases or in a spreadsheet.

Change history

Only Paraguay and Zambia publish a full change history using releases and records. Colombia and Argentina publish the latest information about each contracting process.

Moldova publishes multiple compiled releases about each contracting process. This approach is non-conformant and not supported by OCDS tools or methodologies.

Publishing a change history enables users to track change over time. This is important for identifying inefficiencies and corruption red-flags.

Using data

All publishers have implemented data visualization tools on top of their OCDS data. Providing user-friendly visualization tools helps less technical users engage with contracting data.

Using your own data is good practice since it can help to identify data quality issues and opportunities for improvement. In particular, Moldova's MTender system puts OCDS at the heart of its operation.

Moldova's implementation also demonstrates the value of publishing interoperable data. Their visualization tool combines OCDS data published from two separate procurement systems to give a complete view of procurement in Moldova.

Appendix 1: Comparison table

	Zambia	Paraguay	Moldova	Colombia	Argentina
Source systems	1	1	1	3	3
Contracting processes	617	175,000	60,000	7,000,000	158
Architecture	Middleware with OCDS datastore	Middleware with OCDS datastore	CQRS service orientated architecture, with OCDS-based data models and public query API	Middleware with OCDS datastore	Middleware with OCDS datastore
Extracting data	Event-based (SOAP web-service with async queue)	Pull-based, every 15 minutes (Python, SQL)	Event-based (Camunda, Kafka)	Pull-based, daily (SQL Server 2016 Data Warehouse)	Pull-based, daily (Pentaho Data Integration, Kettle PDI)
Transforming data	Overnight processing (Enterprise JavaBeans)	Every 15 minutes (Python, flatten-tool, ocdskit)	Real-time (Materialized views)	Daily (.Net Core+)	Daily (Pentaho Javascript Node)
Loading data	OCDS records + raw data (MongoDB)	OCDS releases, OCDS records (ElasticSearch)	OCDS-like records (Apache Hadoop)	OCDS releases (MongoDB)	OCDS releases (MongoDB)
Access methods	API, bulk download	API, bulk download	API	API, bulk download	API, bulk download
Publication formats	JSON	JSON, CSV	JSON	JSON	JSON, CSV
Change history	Yes (OCDS records with embedded, compiled and versioned releases)	Yes (OCDS releases and OCDS records)	No (Multiple records per contracting process but does not conform to OCDS)	No (OCDS releases, latest versions only)	No (OCDS releases, latest versions only)
Coverage	planning, tender, awards, contracts	planning, tender, awards, contracts, implementation	planning, tender, awards, contracts	planning, tender, awards and contracts	planning, tender, awards, contracts, implementation
Using data	Visualization and reporting module (MongoDB, Spring Data and D3 charts)	Open-source visualization tool, IADB MapaInversiones tool	Marketplace platforms, internal BI tool, open-source visualization tool	Visualization tool	OC-Explorer portal