# Data Extraction Service Architecture

[Edit](https://github.com/Prifender/anvizent/wiki/Data-Extraction-Service-Architecture/_edit)[New Page](https://github.com/Prifender/anvizent/wiki/_new)

Konstantin Komissarchik edited this page on Feb 13 · [9 revisions](https://github.com/Prifender/anvizent/wiki/Data-Extraction-Service-Architecture/_history)

# **Overview**

The Data Extraction Service (DES) is a component of the Prifender product that handles extracting data from a wide variety of data sources and delivering it for analysis to other Prifender components. The actual data extraction is handled by Talend, an open source ETL framework. The DES is composed of a controller and a set of data extraction nodes.

# **Controller**

This process implements [DES REST Service API](https://github.com/Prifender/anvizent/blob/master/des.api/swagger.yaml) and manages the data extraction nodes. Only one controller is active at a time.

* Must be implemented by extending from the [provided service stub](https://github.com/Prifender/anvizent/tree/master/des.stub)
* Must be a [Spring Boot](https://projects.spring.io/spring-boot/) application
* Must continuously store its state on disk in a JSON file
* Must read a configuration file to get addresses of data extraction nodes

# **Job Compilation**

When the DES controller receives a data extraction job, it will need to compile one or more Talend job that is specific to the target data source. It is expected that when faced with extracting large volume of data, the DES will intelligently break up a single extraction job into multiple Talend jobs to optimize the extraction time.

Adding support for a new data source type will require extending the compiler, but the rest of the DES should not have to change.

[Prioritized List of Data Source Types](https://github.com/Prifender/anvizent/wiki/Data-Source-Types)

# **Encryption of Credentials**

The credentials that DES stores for connection profiles must be encrypted with a key provided to DES. The mechanism for how the key is to be provided is TBD.

# **Data Extraction Node**

The data extraction node receives Talend job requests from the Controller and executes them. It delivers the extracted data to the messaging service. One or more data extraction nodes would be active at any given time, depending on customer data volume and available hardware.

# **Transformation**

After a data row is extracted, one or more transformations to attribute values may need to be applied, as specified in the job definition. Out of the box, DES will come with a set of transformation functions and custom functions can be added to a running system through DES API.

A transformation function is [implemented in Java](https://github.com/Prifender/anvizent/blob/master/des.transform.api/src/main/java/com/prifender/des/transform/Transformation.java). Standard functions are located in [the provided module](https://github.com/Prifender/anvizent/tree/master/des.transform.standard) and are loaded using Java ServiceLoader. See unit tests for loading and usage examples.

# **Messaging Service**

The data read by the data extraction jobs should be delivered to a messaging service, directly from the Talend node doing the extraction. The provided messaging.api module should be used for this purpose, rather than assuming a particular messaging service implementation. Each job should deliver to a separate queue, named after the job id. Each extracted data row gets it's own message, encoded in JSON.

For instance, if a job is scanning a table and extracting data from columns FirstName and LastName, each message written to the corresponding queue should be of the following format:

{

"FirstName": "Joe",

"LastName": "Somebody"

}

# **Docker**

Prifender is deployed using Docker Swarm. DES must be packaged as two docker images. One for controller and one for the data extraction nodes.

# **Resiliency and High Availability**

In case of a VM failure, Docker Swarm will restart the lost containers on another VM. DES must be written such that it can resume operation after the lost containers are recovered.

If the controller is lost, it must be able to resume operation without loosing track of the previously-active jobs or other state.

If a data extraction node is lost, it is acceptable if extraction resumes from the beginning of a chunk or a collection. That is, duplicate data rows are acceptable in case of a VM failure and recovery if there is no efficient method for avoiding duplicates.

# **Future Direction**

None of the following is in scope for M1.

* DES controller should be able to automatically scale up and scale down the number of data extraction nodes (docker containers) that are active based on instruction provided by Prifender central controller.
* We may want to switch from JSON to Apache Avro for message format to optimize perf and on disk footprint. Avro would also make it easier to handle binary data, such as when we scan a file system full of image files.
* DES will need to store configuration and state in a resilient repository (not a JSON file). What repository will be used is yet to be decided. Could be Apache ZooKeeper. Could be something else.
* The Data Extraction Service (DES) implemented the below structured adapters(simplified,relational,hierarchical)
* Struc
* SQL Server
* MySQL
* MongoDB
* Couchbase
* SAP HR (ECC 6.0)
* Hive
* Teradata
* HBase
* HortonWorks2.2
* Azure Data Warehouse
* Sales Force
* DB2
* PostgreSQL
* Cassandra
* SharePoint
* RedShift
* Kafka
* Active Directory
* File System
* Sybase
* Cloudera
* Oracle Enterprise Suite
* Elastic Search 5.X
* Sap - Other Modules
* Sap Hana
* Workday
* CosmosDB
* Google BigQuery
* Net Suite
* DynamoDB
* Informix
* Responsys CRM
* People X
* FieldGlass
* Azure DataLake
* The Data Extraction Service (DES) implemented the below unstructured unstructured.
* 1).box
* 2).google drive.
* 3).unbox.
* 4)confluence wiki.
* 5).imap.
* 6).sharepoint.