Destination Earth Core Service Platform

DESP Monitoring Interface Control Document

Destination Earth
Funded by the European Union
Implemented by ECMWF esa EUMETSAT

Reference: DEST-SRCO-IF-2300369
Date: 18/10/2023
Version:1.1

Role/Title	Name	Signature	Date
Author	Vincenzo Ritrovato		18/10/2023
Verified	Barbara Borgia		18/10/2023
Approved	Andrea Tesseri		18/10/2023

Change register

Version/Rev.	Date	Change	Reason
1.0	14/09/2023		First release of the document
1.1	18/10/2023	Added management of an external service NOT in runtime platform, section 2.9, annex 2 Updated section 2.1	Release to integrate an external service to monitoring

Table of Contents

Index of Figures

Index of Tables

1. Introduction

1.1 Scope

This document is a deliverable in the "*Destination Earth – DestinE Core Service Platform Framework – Platform & Data Management Services*".

1.2 Purpose

The purpose of this Interface Control Document (ICD) is to define the interfaces (external and internal) and data flows of the DESP monitoring, for what concerns the Service Operations Monitoring Dashboard (SOMD), element of the DESP Dashboard Services (please refer to [AD-7]) It has been written following ECSS-M20A standard.

1.3 Applicable Documents

Ref.	Title	Reference and Version
AD-1	[DP-SOW] Statement of Work - Destination Earth – Destine Core Service Platform Framework – Platform & Data Management Services	ESA-EOPG-EOPGD-SOW-10, v 1.0
AD-2	[AD-DSP-TSR] DESP Framework – Platform & Data Management Services – Technical and Service Requirements	ESA-EOPG-EOPGD-RS-10, v1.0
AD-3	[AD-DDL-DP] DestinE – System Framework – Data Portfolio	EUM/TSS/DOC/22/1279455, v1G, 09/09/2022
AD-4	[AD-DSP-SR] DESP Framework – Platform & Data Management Services – Security Requirements	ESA-ESO-SSRS-2022-0111, v1.0
AD-5	Space engineering – Software	ECSS-E-ST-40C, 06/03/2009
AD-6	DESP Monitoring System Design Document	DEST-SRCO-DD-2300376, 1.0
AD-7	DESP SDD and Master ICD	DEST-SRCO-DD-2300317, 1.0

1.4 Reference Documents

Ref.	Title	Reference and Version
RD-1.	DESP Monitoring Data Model	DEST-SRCO-TN-2300345, 1.1
RD-2.	TBD (Documentation API/plugin to retrieve log centralized)	TBD (by TAS)
RD-3.	Grafana Loki documentation	https://grafana.com/docs/loki/latest/
RD-4.	Kafka documentation	https://kafka.apache.org/
RD-5.	Logstash documentation	https://www.elastic.co/logstash

RD-6.	Elasticsearch documentation	https://www.elastic.co/
RD-7.	Kibana documentation	https://www.elastic.co/kibana
RD-8.	Elastalert2 documentation	https://elastalert2.readthedocs.io/en/latest/elastalert.html
RD-9.	Postgres documentation	https://www.postgresql.org/
RD-10.	Filebeat documentation	https://www.elastic.co/guide/en/beats/filebeat/current/index.html
RD-11.	Filebeat installation and configuration	https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-installation-configuration.html
RD-12.	Filebeat output configuration	https://www.elastic.co/guide/en/beats/filebeat/current/elasticsearch-output.html#path-option
RD-13.	Filebeat parsing log file	https://www.elastic.co/guide/en/beats/filebeat/current/dissect.html
RD-14.	HAProxy configuration manual	https://www.haproxy.com/documentation/hapee/latest/onepage/

1.5 Acronyms and Abbreviations

Acronym	Definition
AD	Applicable document
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
DESP	DestinE Service Platform
DLK	Data store
DWH	Data warehouse
ECSS	European Cooperation for Space Standardization
ELK	Elastic Logstash Kibana
ESA	European Space Agency
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
ICD	Interface Control Document
JSON	JavaScript Object Notation
k8s	Kubernetes
RD	Reference Document
S3	Amazon Web Service Storage
SMTP	Simple Mail Transfer Protocol
SOW	Statement of Work
SQL	Structured Query Language
TAS	Thales Alenia Space

2. DESP monitoring overview

2.1 DESP monitoring interfaces design overview

In this Section it is described the DESP monitoring in terms of external and internal interfaces.

Figure 1 depicts the interfaces of DESP monitoring. The only interface towards the 'DESP services to be monitored' is the API/plugin to allow the retrieving of log files from a centralized point, the S3 bucket. Grafana Loki (please refer to [RD-3]) is the technology chosen by TAS to centralize the log files, from stdout/stderr of each service.

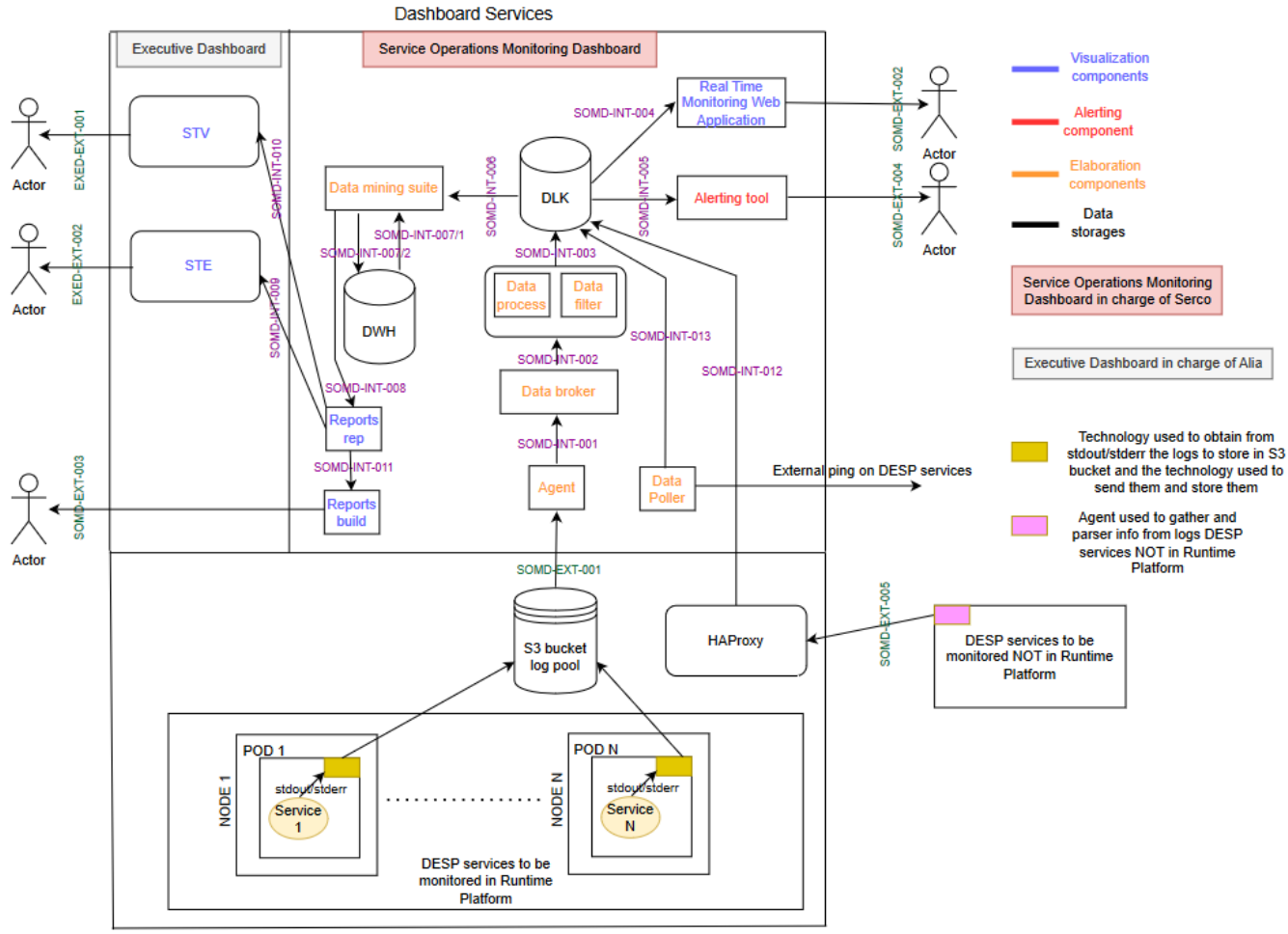

Figure 1: SOMD interfaces

The following table reports the specifications of DESP Monitoring interfaces.

Table 1: DESP monitoring interfaces

Interface ID	From	To	Communication channel	Data Format	Reference

SOMD-EXT-001	S3 bucket log pool	Agent	API/plugin via http	ASCII log file	[RD-2]
SOMD-INT-001	Agent	Data broker	Kafka protocol	ASCII log file	[RD-4]
SOMD-INT-002	Data broker	Data process/Data filter	Logstash protocol lumberjack	ASCII log file	[RD-5]
SOMD-INT-003	Data process/Data filter	Data store	http	Index json	[RD-6]
SOMD-INT-004	Data store	Real time monitoring web application	http	Index json	[RD-7]
SOMD-INT-005	Data store	Alerting tool	http	Index json	[RD-8]
SOMD-INT-008	Data mining suite	Reports rep	Postgres protocol over TCP/IP	Json files Csv files	[RD-9]
SOMD-EXT-002	Real time monitoring web application	user	https	HTML files	
SOMD-EXT-004	Alerting tool	user	SMTP	email	
SOMD-INT-012	HAProxy	Data store	http	Index json	[RD-14]
SOMD-EXT-005	DESP service NOT in Runtime Platform	HAProxy	https	Index json	[RD-10][RD-11][RD-12][RD-13]

2.2 Agent Interface (SOMD-EXT-001)

The interface SOMD-EXT-001 in Table 1 is referred to the retrieving of log files (services logs and k8s cluster logs) from an S3 bucket to send log rows to the chain responsible of buffering, filtering, parsing and indexing them.

2.2.1 Log format

In order to perform the actions on log rows, finalized to index them into DLK (NoSQL database), it is needed to have a row log format containing the right fields to index (please refer to Figure 2). The expected row log format of each log files stored in S3 bucket is the following:

```
[Timestamp:'<DATETIME>' - Level:'<LEVEL>' - Method:'<METHOD>' -
RequestPath:'<REQUEST_PATH>' - Usage:'<USAGE>' - User_token: '<USER_TOKEN>'-
User_id:'<USER-ID>'] <MESSAGE>
```

where the:

- **<MESSAGE>** is the message logged by each service or by cluster log for a specific event
- **Timestamp:'<DATETIME>'** is the time when the event associated to the log row is occurred
- **User_id:'<USER-ID>'** is the actor triggering the event (user or service)

The **<MESSAGE>** must contain all fields requested by monitoring and reporting, listed for each event of each service in DESP monitoring data model (please refer to [RD-1]), excluding event time and user-id, already logged by the other part of log row.

A suggestion to the structure of **<MESSAGE>** of a service could be the following:

[to which is referred the event][event_related_sentence][event_type][event_related_info]

- **to which is referred the event** → for instance a product of Data workflow
- **the specific sentence of the event** → for instance the sentence about product downloaded by Data workflow
- **the event type** → for instance a download of a product on Data workflow
- **all the event related info** → for instance the data provider, the product size and so on of a product in Data workflow

2.2.2 Storing on S3 bucket (retention and organization logs on S3 bucket)

The centralized point is an S3 bucket from which, using the API/plugin (please refer to [RD-2]), the log files are retrieved. These files are organized in order to distinct the source generating the log. The retention applied is 6 months online and 20 years offline (like the security logs).

2.3 Buffering log messages (SOMD-INT-001)

In this section it is described the interface allowing log rows streaming, with technology Kafka [RD-4]. The streaming process captures log rows in real-time from event source (producer like the Agent in Figure 1, implemented with Filebeat [RD-10]), stores these log rows (buffering action) durably for later retrieval, manipulating, processing and routes the log rows to different destination as needed (consumer like Data process/filter in Figure 1). The steaming action is needed to ensure that log messages are not lost when DLK is not reachable.

2.4 Extracting fields and Indexing fields Interfaces (SOMD-INT-002, SOMD-INT-003)

The interfaces SOMD-INT-002 and SOMD-INT-003 are responsible of extracting/indexing fields from log row (please refer to Figure 2). These interfaces are plugin-based in Logstash [RD-5], a component of ELK stack implementing the Data process/filter in Figure 1, and thanks to a wide range of plugins it is possible to collect, process and forward data in many different architectures, configuring ad-hoc the input/filter/output chain of Logstash (input plugins, filter and process plugins, output plugins).

Processing is organized into one or more pipelines. In each pipeline, one or more input plugins receive or collect data that is then placed on an internal queue. This is by default small and held in memory, but can be configured to be larger and persisted on disk in order to improve reliability and resiliency.

Processing threads read data from the queue in micro-batches and process these through any configured filter plugins in sequence.

Once the data has been processed, the processing threads send the data to the appropriate output plugins, which are responsible for formatting and sending data to DLK.

2.5 Monitoring Data visualization (SOMD-EXT-002, SOMD-INT-004)

This section describes the connection between the Real Time Monitoring Web Application (Kibana component of ELK stack [RD-7]) and the DLK storing the log rows indexed (Elasticsearch component of ELK stack [RD-6]). The dashboards on Real Time Monitoring Web Application are retrieved by user with an https secure request and show data (indexes) on DLK provided by an http request on port 9200. The dashboards contain in the detail all metrics useful at operational level to monitor services and k8s cluster.

2.6 Data model DWH (entity E/R model)

TBW *[this section will be written in the next version of the document]*

2.7 Alerting Tool (SOMD-INT-005, SOMD-EXT-004)

In this section it is described ElastAlert2 [RD-8]. The framework will be used for alerting on anomalies, spikes, or other patterns of interest from data in DLK. The near real time data written in DLK can be alerted when these data match certain patterns.

It works by combining DLK with two types of components, rule types and alerts. DLK is periodically queried and the data (stored in an index) is passed to the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match.

This is configured by a set of rules, each of which defines a query, a rule type, and a set of alerts. Several rule types with common monitoring paradigms are included with ElastAlert2:

- "Match where there are X events in Y time" (frequency type)
- "Match when the rate of events increases or decreases" (spike type)
- "Match when there are less than X events in Y time" (flatline type)
- "Match when a certain field matches a blacklist/whitelist" (blacklist and whitelist type)
- "Match on any event matching a given filter" (any type)
- "Match when a field has two different values within some time" (change type)

An example of alert type is the "email" sent to the email address of user by SMTP protocol.

2.8 Data Mining Interface for Reporting (SOMD-INT-008)

In this Section it is described the interface that retrieves the data archived in the long-term Data Warehouse (DWH), which is a relational database implemented using PostgreSQL [RD-9], to extract statistics useful to create reports through ad-hoc queries; database queries are requested by means of SQL scripts. SQL data retrieval queries are used to extract data from a database in a readable format according to the user's request (for instance in json format).

The data extracted from the Data Warehouse are then used in several applications, and, in particular, to make reports like:

- Quarterly Report
- Annual Report
- Weekly Report

2.9 Agent Interface for DESP services NOT in Runtime Platform (SOMD-INT-012, SOMD-EXT-005)

This section describes the interfaces needed to index logs rows into DLK, provisioned by a service not in Runtime Platform. This external service, through an Agent (Filebeat [RD-10]), sends to DLK the info parsed from logs rows, passing through a proxy to manage the access to the DLK from internet (outside of Runtime Platform). The proxy technology used is HAProxy a free, very fast and reliable reverse-proxy offering high availability, load balancing, and proxying for TCP and HTTP-based applications (please refer to [RD-14]).

Annex 1. Procedure to integrate a registered service (on the Runtime Platform) with DESP monitoring

Logs provisioning to DESP monitoring

This section outlines how a DESP registered service deployed on the Runtime Platform can integrate itself with DESP monitoring to have a system allowing monitoring and reporting of the service. In Figure 2 is depicted the provisioning process of log files from the service to the centralized point which is Grafana Loki (please refer to [RD-3]).

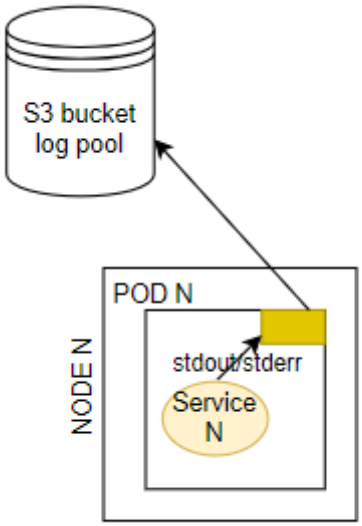

Figure 2: Service logs provisioning

The service will show its log on stdout/stderr of its container. In this way, when the service is started, the stdout/stderr is captured by kubelet agent of k8s which will collect these streams and will provide them to the component responsible to centralize the log in S3 bucket (mentioned previously).

For this scope, the service k8s manifest can contain a tail command like this:

```
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: example
    image: example-image
    args: [/bin/sh, -c, 'tail -f /path-log-example/example.log']
```

Indexing service events into DLK

If a service needs to index some specific events into data store, the relevant Service Provider should request to Serco, having in charge the DESP monitoring, the actions to configure appropriately the Logstash component, to process and filter the service log files and to create a monitoring account to edit/view the dashboards. The DLK data model [RD-1] will be integrated with the events requested by Service Provider. In the following the process to follow:

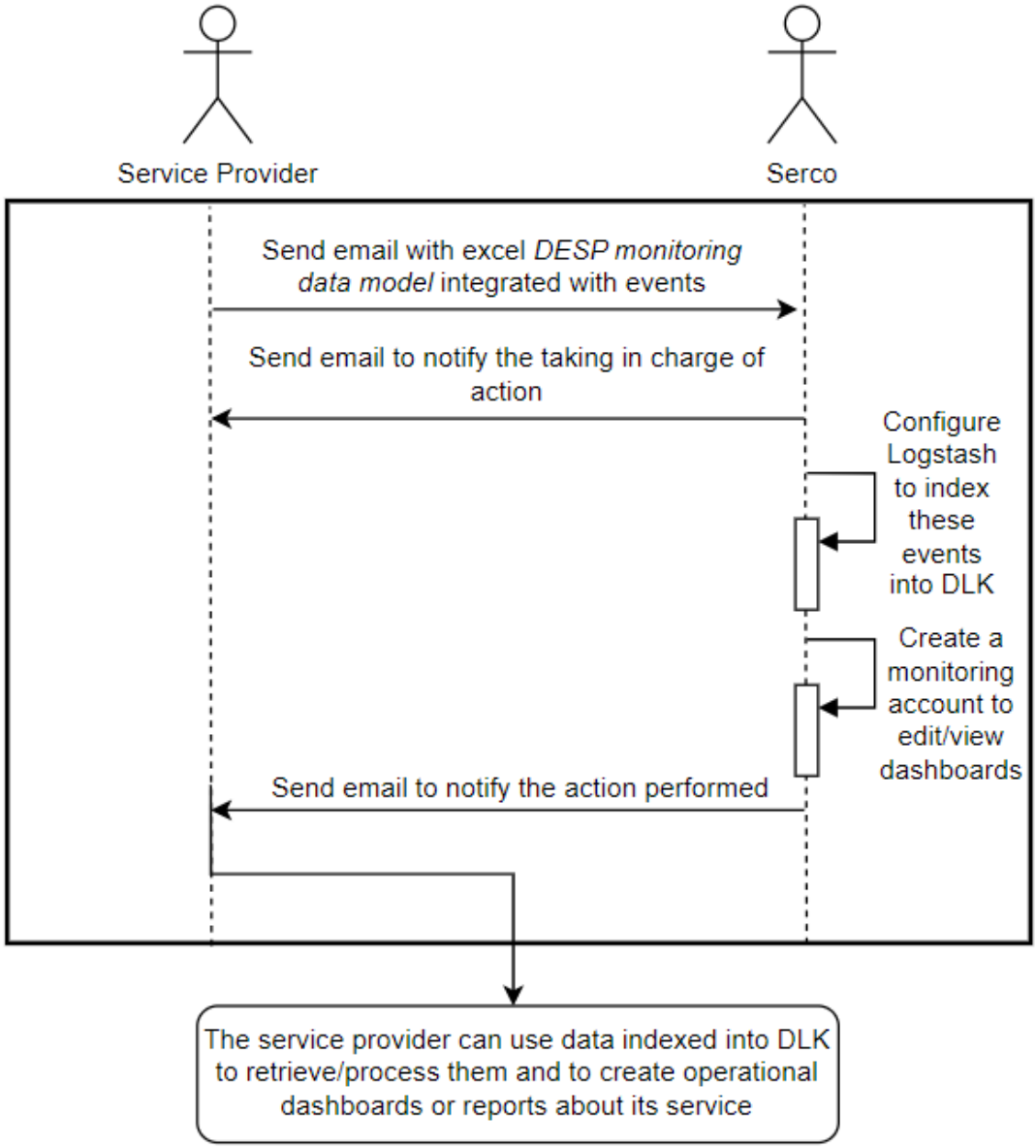

Figure 3: Process to index service events

Annex 2. Procedure to integrate a registered service (NOT on the Runtime Platform) with DESP monitoring

Logs provisioning to DESP monitoring

[Please note that this proposed procedure is currently under evaluation]

In this section it will be explained how a DESP registered service not deployed on the Runtime Platform, can integrate itself with DESP monitoring.

In the machine where this service runs, the Service Provider has to install the Agent Filebeat and must configure the filebeat.yml file in order to parse logs applications and to send data to DLK through a proxy.

In order to install the Agent please refer to [RD-11].

The output setting of Agent is specified in Elasticsearch section in filebeat.yml by the parameter proxy_url [RD-12], where this url is provided by Serco.

The parsing of logs is performed following the guide lines configuration of filebeat.yml in [RD-13].

The Service Provider must request Serco a monitoring account to edit/view dashboards with own data indexed.