Distress Tracking Data Repository (DTR) Architectural Considerations

Version May 21st, 2019

Executive Summary

This document contains architectural considerations about the Distress Tracking Data Repository (DTR). These considerations take into account ICAO System Wide Information Management and related service orientation to form a baseline in support of developing a DTR solution. The DTR functional requirements are reviewed and possibilities for information services, technical infrastructure and system architecture are discussed. The primary users of the DTR are ATSUs and RCCs. The DTR, which following the DTR functional specification is a data persistency layer, is described as a store of last known position data for aircraft in distress. Data format options are discussed including the use of Open Geospatial Consortium standards and related technologies to provide the DTR with globally interoperable functionality. The provided considerations form a foundation for the remaining technical system architecting and development work by the actual provider(s) of the DTR.

Contents

1. General

1.1. Introduction

The architectural considerations in this document are exclusively based on the GADSS concept [Ref 01] and the DTR functional requirements [Ref 02].

1.2. Disclaimer

This document cites technologies and vendor solutions for illustrative purposes only. Consequently, these citations do not indicate any preference nor does this exclude any other vendor solution or technology that may not have been cited.

1.3. Scope and document purpose

Within the wider GADSS concept [Ref 01], the DTR is a sub-system that stores, retains and shares the last known position information of aircraft in distress. This document builds on the DTR functional requirements [Ref 02] to provide a foundation for the remaining technical system architecting and development work by the actual provider(s) of the DTR.

According to the DTR functional specification [Ref 02], inputs to the DTR are provided by "contributors" and outputs of the DTR are consumed by "users". This document includes considerations on how DTR contributors provide information to the DTR, and how to ensure that DTR users access and make good use of DTR information in a diverse technology landscape world-wide.

The DTR related information exchanges are expected to be based on the emerging ICAO System Wide Information Management (SWIM) principles, in other words applying service oriented architecture principles. Consequently the system interfaces to be provided as information services are covered in the document.

The DTR system scope addressed in this document is depicted in **Figure 1**:

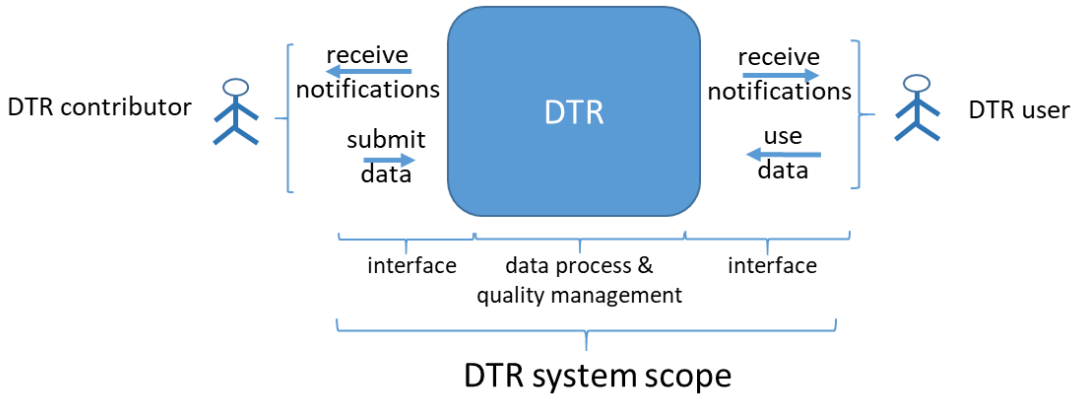

Figure 1: DTR system scope

To build upon concrete industry standards, the architectural considerations provided are based on the EUROCONTROL SWIM Specifications [Ref 04][Ref 05][Ref 06], in accordance with ICAO SWIM as described in ICAO Doc 10039 [Ref 03].

1.4. Terminology

[TBD]

1.5. Acronyms

- ATSU: Air Traffic Service Unit
- ADT: Autonomous Distress Tracking
- AIRM: ATM Information Reference Model
- AMQP: Advance Message Queueing Protocol
- API: Application Programming Interface
- ATS: Air traffic service
- DCIS: Data Contributor Information Service
- DTR: Distress Data Tracking Repository
- DUIS: Data User Information Service
- GADSS: Global Aeronautical Distress & Safety System
- GeoJSON: Geography Java Script object notation
- GIS: Geographic Information System
- GML: Geography Markup Language
- GUI: Graphical user interface
- HTTP: Hypertext transfer protocol
- IER: Information Exchange Requirements
- IP: Internet protocol
- JSON: Java Script Object Notation
- KML: Keyhole Markup Language
- LDAP: Lightweight directory access protocol
- MEP: Message Exchange Pattern
- OASIS: Organization for the Advancement of Structured Information Standards
- OGC: Open GeoSpatial Consortium
- QGIS: Quantum GIS
- RCC: Rescue Coordination Centre
- REST: Representational State Transfer
- SAR: Search and Rescue
- SOAP: Simple Object Access Protocol
- SWIM: System Wide Information Management
- TCP: Transmission control protocol
- TI: Technical infrastructure
- TLS: Transport Layer Security
- URL: Uniform Resource Locator
- WFS: Web Feature Service
- WMS: Web Map Service
- WS-N: Web Services Notification
- XML: Extensible Markup Language

1.6. References

- [Ref 01] ICAO GADSS CONCEPT

- [Ref 02] ICAO DTR Functional specification
- [Ref 03] ICAO Doc 10039 – Manual on SWIM concept.
- [Ref 04] EUROCONTROL SWIM specification: Technical Infrastructure Yellow Profile
- [Ref 05] EUROCONTROL SWIM specification: Information Definition
- [Ref 06] EUROCONTROL SWIM specification: Service Description
- [Ref 07] AMQP1.0: http://www.amqp.org/
- [Ref 08] OASIS Web Services Notification : http://www.oasis-open.org/
- [Ref 09] OpenLayers: http://openlayers.org/
- [Ref 10] OGC WFS: http://www.opengeospatial.org/standards/wfs
- [Ref 11] OGC Publish/Subscribe: http://www.opengeospatial.org/standards/pubsub
- [Ref 12] OGC Publish/Subscribe SOAP binding: http://www.opengeospatial.org/standards/pubsub
- [Ref 13] GeoServer: http://geoserver.org
- [Ref 14] OGC Testbed Engineering Reports https://www.opengeospatial.org/docs/er
- [Ref 15] FIXM – www.fixm.aero
- [Ref 16] AIRM – www.airm.aero

2. DTR Functional requirement overview

The complete set of DTR functional requirements is available in the DTR functional specification [Ref 02]. A selection of key aspects is depicted in this section to develop an overview of the DTR in relation to the architectural considerations to make. **Figure 2** below provides the highlights from a contributor and user perspective.

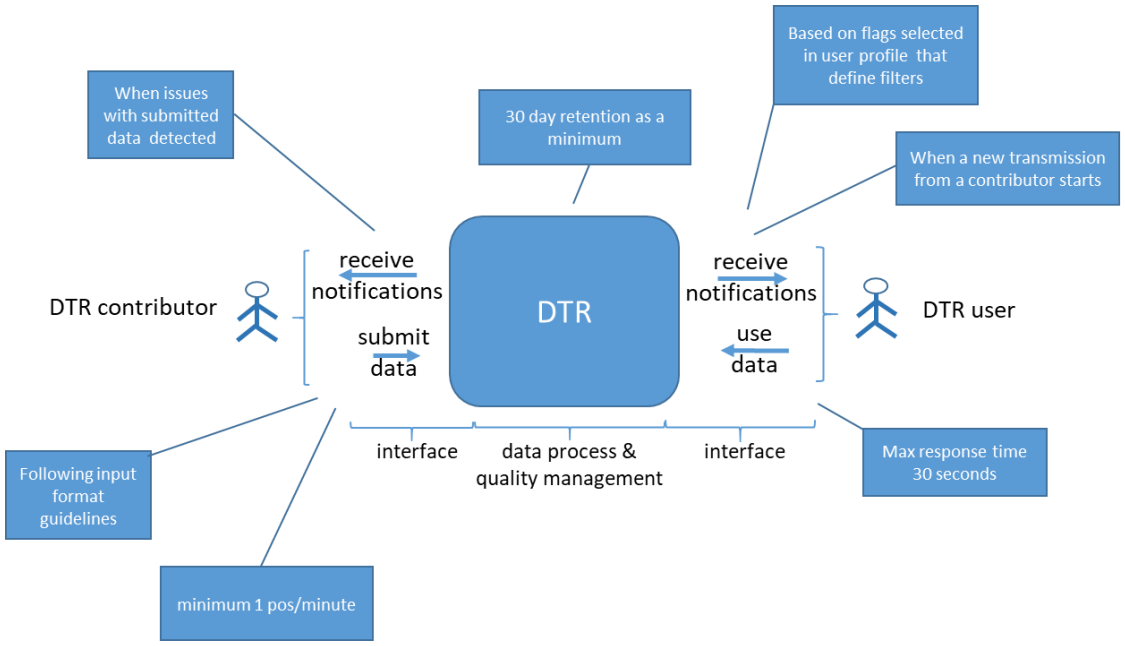

Figure 2 : DTR functional requirements highlights

2.1. Roles

The DTR functional specification identifies the following roles:

- DTR contributor
  - is a user of a State approved ADT;
    - Note: it is not excluded that a DTR contributor is also the DTR host (an ADT service provider hosting the DTR system)
  - is accredited based on accreditation requirements;
  - is able to comply to requirements of DTR;
  - is a “developmental submitter” when not yet accredited but may use the DTR development/test system for testing.
- DTR user
  - is an individual or entity using the DTR;
  - is accredited based on accreditation requirements;
  - has read access to DTR data based on user profile;
  - may have multiple users and at least on super user per entity;
  - uses filters to define user entity profiles.
- DTR host
  - is hosting the DTR system by delegation;
  - the role is performed on a 24/7 basis;
  - requires established SLAs.
- DTR administrator

  - ICAO is the attributed DTR administrator role;
    - Note: ICAO is nominally the administrator but may delegate
  - the role is performed on a 24/7 basis;
    - Note: the response to general queries and accounts (Mon-Fri 9-5 ICAO HQ Montreal business hours) and ensuring access to the DTR (24/7)
  - the DTR administrator establishes criteria to approve DTR contributors.

2.2. Business rules

The DTR functional specification identifies the following business rules:

- Following accreditation, every DTR contributor obtains a user account from the DTR administrator;
- Following accreditation, every DTR user obtains a super user account from DTR administrator;
  - the super user adds user accounts in his/her own “user entity” remit;
  - the super use manages user type profiles including criteria and options;
  - the user sets notification flags within the bounds of the user type profile received;
- DTR contributors have read/write access on provided data;
  - Every DTR contributor is also a user of the DTR;
- DTR users have read only access according their user profile.

2.3. DTR Notifications

The DTR functional specification identifies the following DTR notifications:

- DTR host to DTR contributor
  - Rejected message notification
  - Incomplete message notification
  - Duplicate message notification
- DTR host to DTR user
  - Activated ADT data available notification (Only on first message received following activation.)
- Notification delivery mechanisms
  - Email
  - SMS
  - ATS message over AFTN
  - Message queueing service

The figure below depicts roles, business rules, notifications, DTR system components and interactions:

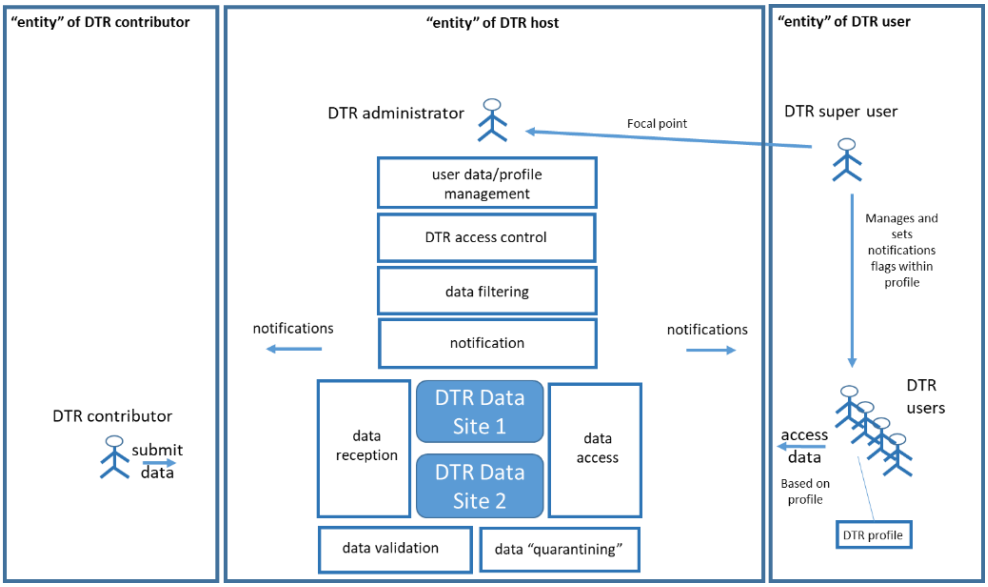

Figure 3: DTR system components highlights

As shown in **Figure 3** above, different roles participate and interact in the DTR operation in order to receive, retain and make accessible last known position data. The DTR needs to inform the actors when necessary using a notification mechanism and a data filtering mechanism is foreseen based on flags set in the user profiles.

Moreover, a key upfront functional requirement is that DTR data is replicated in at least two stores on geographically disjoint locations. "Geographically disjoint" in this context should be understood as data to be replicated on 2 or more systems that are geographically and physically disjoint, the latter implying another machine hosting. Hence, the DTR may not be built on two virtual machines running on the same server(s). The DTR needs to foresee this data replication capability explicitly including the possibility to add data servers when this is required. This makes it clear where the data is actually stored and replicated.

It is furthermore not excluded that the DTR host role could be performed by an entity which is also DTR contributor. At least one DTR contributor and at least one DTR host should exist.

3. Key aspects impacting the architecture

Analysing the DTR functional specification [Ref 02], it could be concluded that some of these requirements may result already in upfront design choices:

3.1. One DTR host and two data replicas

According to the DTR functional specification [Ref 02], the DTR data shall be replicated into at least two physical locations that are geographically disjoint.

It is therefore understood that the DTR needs to be implemented as a single system, i.e. as a "capability in the middle", thus a shared resource for DTR contributors, DTR users and the DTR administrator.

The fact that the data is replicated into a second data store should be transparent to the DTR contributors and DTR users, hence this replication is considered a back-end data management aspect. It is furthermore understood that this architecture decision is perceived to be an added value in providing a "one-stop-shop" for users, however noting that this also implies a single point of failure.

Note: Appendix C contains considerations in the event that multiple DTRs would be built to support the global operations of last known position data.

3.2. Event-driven architecture

Citing Wikipedia, "Event-driven architecture, is a software architecture pattern promoting the production, detection, consumption of, and reaction to events." As stated in the GADSS: "*To identify a distress condition, the aircraft state will be analysed in real-time by aircraft systems or ground processes and the use of event detection and triggering criteria logic will initiate the notification of the alert to assist locating the aircraft in distress.*"

Following the DTR functional specification [Ref 02], alerting is out of scope of the DTR, but it is still a distress event that triggers the distress data exchange and related notifications when required. Consequently, events and the handling thereof are of importance to the DTR system. Although the event distress detection is outside the scope of the DTR, the DTR needs to able to react on inputs and enable downstream access in a way that is sufficiently dynamic.

3.3. DTR in a service-oriented environment

The DTR intends to use System Wide Information Management (SWIM) as the "conduit" to exchange data. In the context of the DTR architecture design, "to use SWIM" means to take into account the key components (i.e. information, information services, and technical infrastructure) of SWIM.

The DTR aims at providing a "globalised solution" which however needs to be used in a localised way, i.e. taking into account to be effective a vast array of local realities and capabilities. As a consequence, to ensure the added value of the DTR, the ability for this diverse array of DTR users to access/consume the information is key to the design of the solution.

This is where the use of SWIM provides added value in terms of providing standard interfaces and "equalized SWIM information", but not necessarily reducing the outward facing interfaces of the DTR to a single solution.

To reason about interfaces between systems, the notion of information service is used, a type of service that provides an ATM information sharing capability, hence including specification of interfaces, MEPs, data format, quality of service, etc.... Building a system that "uses SWIM" therefore means to define and implement information services.

In the remainder of the document, all considerations related to information exchange and how to architect, design and implement these information exchanges are SWIM based as described in ICAO Doc 10039 [Ref 03].

More background information on SWIM is provided in Appendix A – Essentials on SWIM.

3.4. Notifications, data exchanges and messages

In this document context the notion "message" means a unit of communication between two interacting systems. Messages are exchanged between systems when they interact (e.g. when an information service is invoked). The notion "message" does not imply any other connotation that may exist in the current ATM system (e.g. when using store and forward methods to send information over AFTN).

A notification message has a function which is "to notify" (e.g. when "new data available").

A data exchange message has a function which is to "deliver data".

The following DTR interactions involve message exchanges:

3.4.1. DTR contributor interaction

3.4.1.1. DTR contributor workflow

- Obtain a DTR user account following accreditation;
- Subsequent interactions envisaged:
  - Send data;
  - Receive notifications.

3.4.1.2. Send data

DTR contributors require a client application to push (i.e. "write") data to the DTR data store. The application uses a one-way "push" MEP for the purpose.

The DTR contributor event logic that triggers data exchange messages is outside the scope of the DTR.

The DTR host implements an input handler to write received data into the DTR data store.

The related DTR host interface is inward facing. When based on SWIM, both the contributor client application and the DTR host input handler exchange information using a SWIM information service which defines and implements the agreed technical infrastructure interface bindings (technology protocols to be used) and data format of the message payloads.

The main contributors according to the GADSS (see Appendix A in GADSS Concept [Ref 01]) are the involved ADT service providers, operators, CosPas SARSAT, and the involved ATSUs.

3.4.1.3. Notifications

The DTR needs to send notification messages to DTR contributors regarding the reception of the data sent. The DTR host uses a notification handler to send notifications using a one-way “push” MEP. The DTR notification triggers are based on data message handling aspects (e.g. errors). The DTR notification handler needs to know to whom to send notification messages (i.e. when using for instance e-mail as a delivery method).

Given the type of notification messages expected these could be handled at the infrastructure level as part of the nominal “error handling” communications during data exchange activity. This has the advantage that potentially less additional logic needs to be considered at the DTR host level and also that the notification is directly addressed to the data message sender address (i.e. no need to know to whom to send the notification). [Discussion Topic: see 8.4]

3.4.2. DTR user interaction

3.4.2.1. DTR user workflow

Conceptually the complete interaction is from a user point of view based on a publish/subscribe pattern.

- Obtain a DTR user account following accreditation;
- Select subscription options (i.e. flags);
- Subsequent interactions envisaged:
    - Receive notifications;
    - Request data.

3.4.2.2. Notifications

The DTR needs to send notification messages to DTR users when a position update was received following activation of an ADT. The DTR notification is based on DTR user profile setting (flags).

A DTR notification handler acts as an event service that triggers notifications based on subscription matches using a one-way “push” MEP. The DTR notification handler needs to know to whom to send notification messages.

3.4.2.3. Request data

DTR users require a client application to pull (i.e. “read”) data from the DTR store. The application uses a Request/Reply MEP to access the data.

The DTR host implements a query handler to reply to data requests. The related DTR host interface is outward facing. When based on SWIM, both the user client application and the query handler exchange information using a SWIM information service which defines and implements the agreed technical infrastructure interface bindings (technology protocols to be used) and data format(s) of the message payloads.

Since the DTR user launches the request to the DTR (i.e by means of a request message to the known IP address of the DTR) the IP address to which the DTR responds (i.e. by means of a response message) is handled at the connection level of the underlying protocol.

3.4.2.4. DTR user application logic

DTR users need to devise a polling strategy to send their requests. The DTR user application logic may involve more or less automation. It is up to the DTR user client application to put in place application logic that uses both notification and data messages in a coherent way (i.e. establishing a workflow at the DTR user side). To facilitate this automation it could be considered to make the notifications computer interpretable, which is an aspect currently not covered by the DTR functional specification. [Discussion Topic: see 8.4]

3.4.3. DTR "store in the middle"

Based on the discussion above, the DTR is based on a "store in the middle" configuration which uses no specific messaging middleware to distribute data. The DTR host does not push any last known position information out. The configuration is summarized in the following figure:

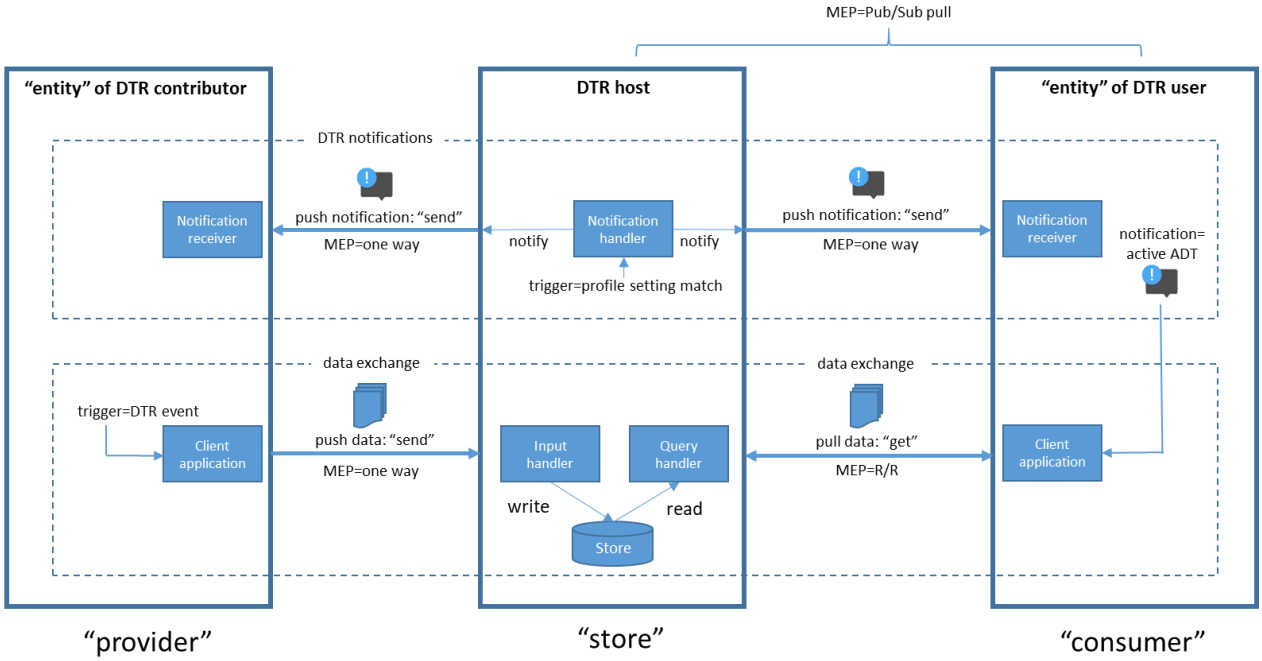

Figure 4: "store in the middle" DTR user view

From the DTR user viewpoint, the overall MEP is a Publish/Subscribe pull. The DTR host notifications trigger in some way DTR user client application polling. The DTR user client application uses a Request/Reply MEP to access data (pull). The DTR contributor client application uses a one way MEP to send data (push). **Figure 4** above abstracts the notification from DTR host to DTR contributor. It should be clear that this "notification" could be implemented as part of the "error handling" at the infrastructure level (see 3.4.1.3).

3.5. Data scope and data definition

The mandatory (M) and optional (O) data envisaged by the DTR functional specification [Ref 2] are:

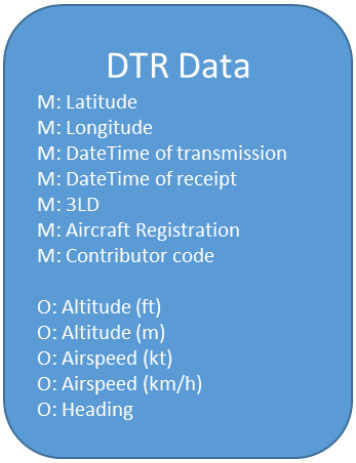

Figure 5: DTR data

The primary users of the DTR information are ATSUs and RCCs. The objective of the DTR is to support emergency situations including Search and Rescue (SAR) missions with last known position information of an aircraft in distress or worse. Considering the global coverage of SAR missions the DTR position information, geo-temporal by nature, should be globally usable, easy to use and have a very high likelihood of successful coding and decoding of the information when exchanged.

It is therefore recommended to consider in the DTR architecture the use of existing globally applicable standards for geospatial information. Note: OGC provides implementations specifications at various levels:

- the data to be exchanged (i.e. the encoding of the data);
- the technical infrastructure interfaces (i.e. the protocols used to perform data exchanges).

Furthermore, since it also important to achieve interoperability between all DTR stakeholders the AIRM [Ref 16] is used as the reference for the information to be exchanged. See Appendix B for draft DTR data mapping and examples for data coding.

For the DTR input side (DTR contributor) good controls for data input assurance purposes are required. In this case it will be appropriate to define a single input format to support that objective.

For output purposes (DTR user), taking into account maximisation of usage and likelihood of successful data consumption, the objective should be to technically support a wide array of applications, anywhere anytime, possibly including proprietary systems, mobile applications both in an ATSUs and RCC/SAR units context. This implies providing interfaces and formats which facilitate data transformations. [Discussion Topic: see 8.1]

3.6. DTR performance

The DTR contributor output stream has a minimum requirement in terms of data provision (number of positions sent per minute) but does not set an upper limit. In general peak load performance criteria should be made explicit so to capture more boundary conditions in which the DTR host has to be operated. [Discussion Topic: see 8.2]

4. DTR architecture components

In this section the DTR architecture is outlined in high-level terms, the main aspects relevant at this stage of DTR development are shown.

4.1. Assumptions based on DTR functional specification

The following assumptions are deemed true:

- All DTR functional specifications remain applicable independently from the level of details provided in this document;
- Single DTR provider;
- At least two data store replicas;
- User and contributor accreditation outside DTR scope;
- Using SWIM information services any accredited DTR contributor, or accredited DTR user can provide data to or consume data from the DTR host;
- Every DTR Contributor is also a DTR User.

4.2. Functional areas of the DTR

Based on the DTR requirements the following functional areas and roles emerge (non-exhaustive picture):

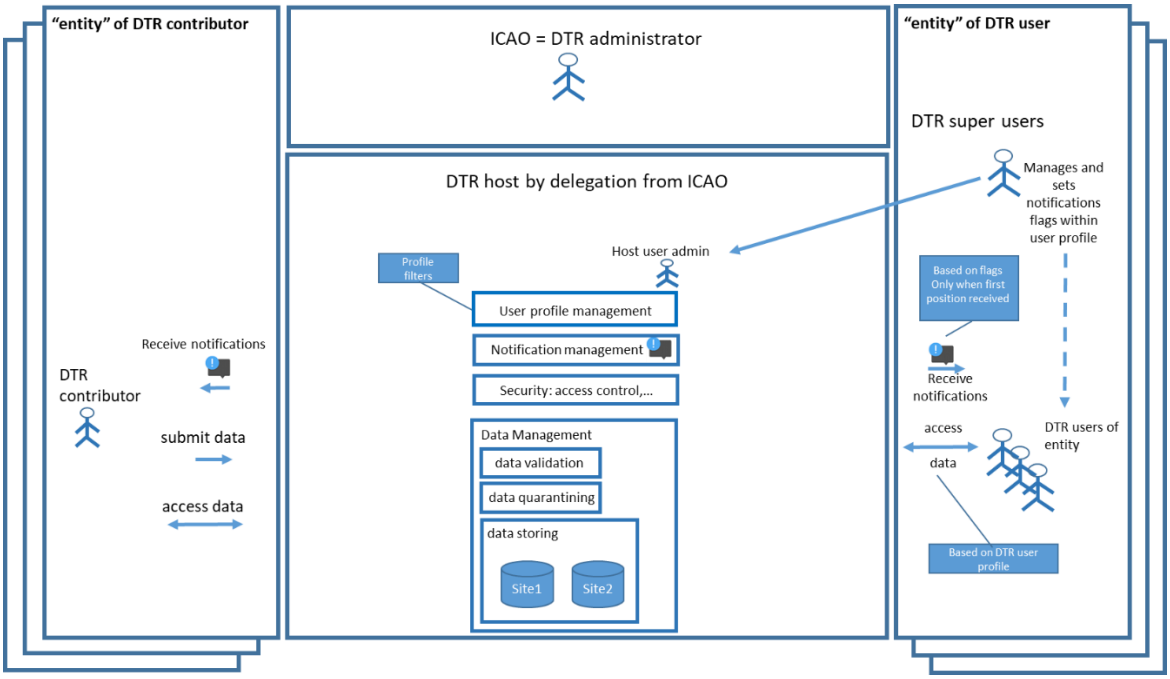

Figure 6: DTR functional areas and roles

4.2.1. User profile management

User profile management ensures user profile data: user accounts, user privileges, user data protection, user subscription etc… This aspect is in the remit of the DTR administrator (ICAO) and is to be based on the envisaged capabilities. A commonly used data exchange protocol is the light-weight directory access protocol (LDAP). It should be noted that some solution environments come

packaged with basic user profile management capabilities. The ICAO OPS Control Directory will be expanded to provide this functionality.

4.2.2. Notification management

Notification management is understood to cover notification of DTR activity to contributors and users. According to the DTR functional specification this can be achieved using means such as, e-mail, SMS, ATS-message (AFTN), message queue service.

Section 2.3 lists the types of notifications envisaged.

Section 3.4 expands on notification in relation to data exchange.

No specific technical requirements are available for the implementation of notifications.

Additional observations:

- Whilst the DTR specification mentions the notion of “message queueing service” as delivery method, no further elaborations on the use of a messaging middleware oriented solution is included having taken into account the overall intended function of the DTR. [Discussion Topic: see 8.4]
- It is deemed valuable to extend notification to include ‘message reception acknowledgements’ notification. There could be an operational significance in terms of “knowing that they know”. [Discussion Topic: see 8.4]

4.2.3. Security management

According to the DTR functional specification, security management is providing access control based on user profiles and single sign on should be followed as a principle.

User authentication and access control are not specific to the DTR. It is assumed that the DTR functional requirements provide sufficient details to implement a solution already existing or readily available from the market.

Security requirements of the DTR in relation to the data stores are: database security, intrusion detection, denial of service protection, virus protection, password encryption, intrusions detection, DTR access restriction.

Secure data exchange is intrinsically part of the SWIM TI interface binding chosen which includes specific security related protocols (e.g. use of TLS for encryption). This implies that for instance HTTP based exchanges shall be secured HTTP exchanges over HTTPs.

In addition, security should be considered by design in the implementation solution used. Encryption of data, certainly in a cloud-based solution, whereby the cloud host may or may not be able to actually decipher the data are aspects of consideration. Moreover, handling of “security zones” in for example a file sharing solution may come ‘out of the box’.

4.2.4. Data Management

Data management encompasses the storing of the ADT position information and its replication throughout associated system (components). In addition, it includes data validation and quarantining.

Validation is considered as the checks performed on the input data when storing. The DTR functional requirements indicate the necessity to foresee a testing system which allows potential DTR contributors to test their interfaces in advance of being accredited based on the verification of their implementation. Validation in this case is done at design time when a contributor implements data provision systems and interfaces.

Validation at run-time is deemed a possible secondary level check assuming that providers employ data management practices and ensure data quality. Hence, once the verified system is running, the QMS should ensure that it performs and keeps performing as designed.

Quarantining implies permanent checking of inputs. Further consideration should be given on the likelihood that this occurs versus the costs related to quarantining. Can for instance the ADT emit incomplete data? Is it then not for the contributor to take measures? [Discussion Topic: see 8.5]

It is assumed that the only reason that incomplete data could be a factor is related to a transmission issues between contributor and DTR host. This aspect will need to be handled at the technical infrastructure level and should be further considered when detailed implementation choices are made. From a conceptual perspective, it could also be more appropriate to operate regular tests in which data contributors send test data to a validator and through this means ensures the required quality of service.

Furthermore, quarantining is also understood to be a data staging in case of transmission problem detected at the technical infrastructure level. It should be noted that regular testing should be considered because the nominal state of the DTR will be idle due to the expected low frequency of distress activations. It means that the DTR will have to switch from long periods of inactivity to sudden situations of “full performance and availability”. Processes on regular testing, including controlled “fake message loads”, should be considered.

5. DTR Service Orientation focus

When considering a service oriented approach for the functional decomposition of the DTR, it essentially means adding inward and outward interfaces (i.e. 'services') that communicate over the required technical infrastructure. The requirements and design aspects of the interfaces are subject to the architecture considerations in this document. The following picture show the DTR in relation to its interfaces:

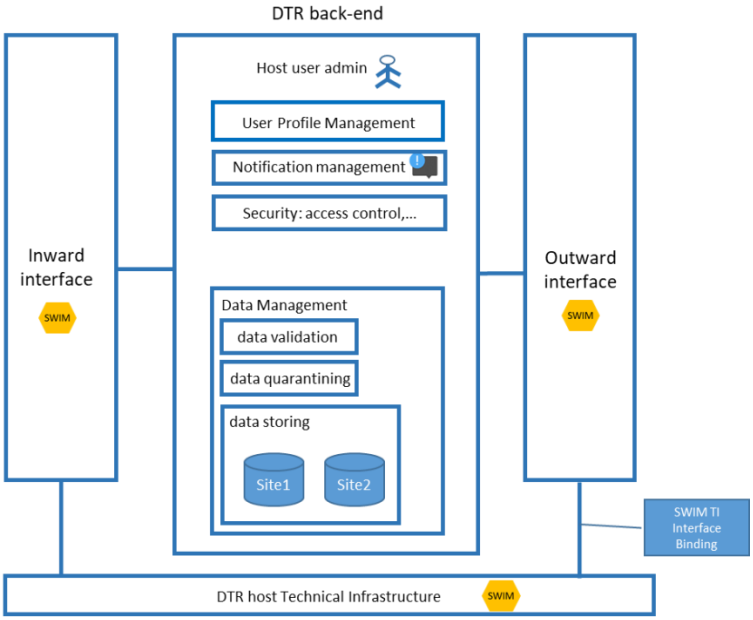

Figure 7: DTR host functional areas and 'services'

In developing the interfaces of the DTR system, these can take either of the following forms:

- information service;
- web application used from within a web browser of a contributor/user/host;
- mobile application.

To consume an information service a consuming client needs to be built. This may be done by for example customizing an application (e.g. QGIS open source as a "thick client"). When using a web application the application can be used via a web browser (i.e. "thin client" approach).

The DTR architecture should lead to a balanced mix that caters for the interests of contributors, users and hosts. These interests are not necessarily the same. A DTR user who consumes an information service to access DTR content would have to do this through a SWIM-enabled client application. Some DTR users may decide to create/adapt an application to consume the DTR host information services directly, others may prefer to use an out-of-the-box solution to consume the information. The DTR architecture considerations provided describe these options which are not mutually exclusive.

The advantage of a service oriented approach for the DTR is that the DTR exposes information services which can be picked up by solution providers to create (third party) custom solutions for the consumer(s) of the information. The focus of the DTR implementation architecture shall be on the interfaces that constitute the boundary conditions to interact.

Data exchange is performed using information services at the application level:

5.1. Specifying information services

When specifying information services in a service-oriented environment, typical questions emerge such as:

- What are the information exchange requirements (IER)?
- Which information is exposed as an information service?
- Which interfaces and operations should the information service expose?
- Which message exchange patterns (MEP) are to be used?
- Which information service is inward facing (write capability) and which ones are outward facing (read capability)?
- Which technology protocols to use to implement interfaces (e.g. SOAP, XML, OGC WFS, HTTP/JSON, AMQP 1.0)?
- What is the format of the message payloads?

In the context of the DTR and based on the DTR functional specification the following facts are known:

- Information exchange requirements (IERs);
- Inward facing and outward facing aspects;
- Overall publish/subscribe approach.

What is not known at this stage:

- Which information services;
- Which protocols to use.

5.2. Data contributor information service (DCIS)

The inward facing data exchange interface is the **Data Contributor Information Service** (DCIS). It performs data inputs from the DTR contributor into the DTR store. The DCIS exposes to the Data contributors a data input interface based using a prescribed encoding for the data. This contains the cost to perform data input.

Multiple contributors (e.g. operator versus ATSU) can be active on the same distress situation. The contributor records the date and time of transmission which is used to determine the last known position, i.e. the sequencing of the records/messages within the DTR for downstream consumption. [Discussion topic: see 8.3]

DCIS design drivers:

- Inward facing to DTR host (“write”)
- Allocated to the DTR host system
- Available 24/7
- Used by
    - DTR contributor to send data
    - DTR host to receive data

    - Possibly also to fulfil DTR host to contributor notifications as part of the interface implementation
- Capable of mandatory and optional data handling
- Capable of handling required data transmission volume peak
    - Peak TBD in accordance with performance requirements
- Message exchange pattern: One Way
- Information:
    - data format: uses a “single format in” approach
    - good geospatial handling
    - data definition: to be determined and traced to AIRM [see Appendix B]
- Technical infrastructure binding
    - Message exchange protocol between DTR host and DTR contributor
        - To be determined (e.g. OGC WFS-T)

Note: Detailed information service design can be done after implementation architecture decisions are made.

5.3. Data user information service (DUIS)

The outward facing data exchange interface is the **Data User Information Service** (DUIS). It performs data outputs from the DTR store to the DTR users. The DUIS possibly exposes more than one data format to maximize user decoding chances and alleviate possible data transformations burden costs the DTR user may have. This constitutes an added value of the DTR host; the DTR host is not only a “one-stop-shop” for last known position data.

DUIS design drivers:

- Outward facing from DTR host “read”
- Allocated to the DTR host system
- Available 24/7
- Used by
    - DTR host to distribute data
    - DTR user to access data
- Capable of mandatory and optional data handling
- Capable of handling required data transmission volume peak
    - Peak TBD in accordance with performance requirements
- Message exchange pattern: Request/Reply

        - When OGC WFS used Synchronous Request/reply
        - In combination with user subscriptions aspect and notifications the overall pattern is Publish/Subscribe Pull
- Information:
    - data format: possibly using a “multiple format out” approach (e.g. based on Open Layers supported formats: GML, GeoJSON, KML, and GeoPackage)
    - good geospatial handling
    - data definition: to be determined and traced to AIRM [see Appendix B]
- Technical infrastructure binding
    - Message exchange protocol between DTR host and DTR user
        - To be determined (e.g. OGC WFS/WMS)

Note: Detailed information service design can be done after implementation architecture decisions are made.

5.4. Data validator service

The data validator function could be exposed as a service. A DTR contributor could submit a DTR data set at design time for testing purposes. Validation can be treated as a DTR back-end function to start with and could be exposed as a service in a latter development phase.

Eventual service design is to be accomplished once overall directions of the architecture is decided upon.

6. Other DTR interfaces

Other interfaces could be considered to be provided as part of the DTR architecture:

- The DTR could expose additional application level tools:
  - DTR GUI as a web application (i.e. dynamic web page)
  - DTR as a mobile application
- The DTR could expose additional static DTR content deemed necessary and useful (e.g. global coverage of FIR/UIR layers, airport locations, waypoints, etc…).
- Capability to download data from the DTR in various formats

The above aspects may lead to a stepwise development of the DTR capabilities when deemed necessary. Well known mapping components can be used: Open Layers, major mapping APIs, GIS data conversion libraries.

6.1. DTR Graphical User Interface (GUI)

Used by:

- DTR host to expose functionality
  - Essentially views on the data: table view, map view (e.g. based on Open Layers), notification view.
- DTR user and contributor as out-of-the box thin client used in a web browser context.

The DTR GUI can itself consume information services. The DTR user does not need to implement a consuming client. The interface takes the form of a web application with a GUI (e.g. dynamic web page).

6.2. DTR Mobile APP

Used by:

- DTR host to expose functionality;
- DTR user as mobile APP.

6.3. DTR user profile administration interface

Used by:

- DTR contributors to manage own user profile data;
- DTR users to manage own user profile data;
- The DTR administrator/host to collect user profile data;
- The DTR administrator/host to manage the subscriptions.

The interface can take the form of a web application.

User data protection is in scope.

6.4. Consuming client application

A consuming client application “MyOwnApplication” is an application which is adapted or built to consume information services available from the DTR host.

- For mapping purposes, GIS viewers that are OGC WFS/WMS capable could link to the information services when based on OGC. More advanced GIS Open Source applications such as QGIS could be used to link and create dedicated map views. Obviously, since the communications are in that case over secured HTTPs this may require some additional customisation.

7. DTR architecture description

7.1. Introduction

The DTR architecture description is in support of further orienting DTR designs. A selection of core elements of the system are shown in a view, completeness not being the objective. The result forms the baseline for further elaboration of the detailed architectural description and specification of building blocks needed for system implementers.

The description provides:

- mapping of functional components into system components;
- mapping of interfaces to information services;
- mapping of interfaces to applications.

In the remainder of the document the following applies:

- “admin data” = the data about contributors and users which is held as part of the DTR user profile management and leads to managing subscriptions;
- “operational data” = the actual DTR data provided by DTR contributors and exploited by DTR users.

Devising the security aspects is considered common to all architectural views. Security, involves access management, authentication, and secure transport as part of the technical infrastructure. This aspect is not further elaborated.

Following symbols are used:

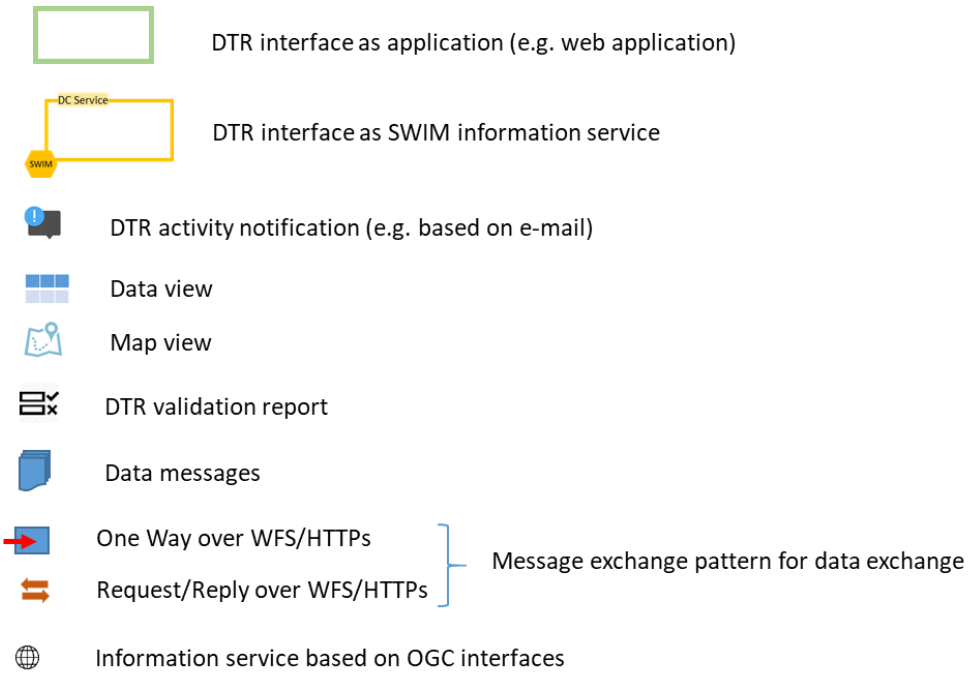

Figure 8: symbology used in architecture views

7.2. DTR architecture view

The following is an elaboration of the “store in the middle” (see section 3.4.3). The architecture uses a GIS server application as the main application environment component in the DTR. An indicative examples of a GIS server applications is GeoServer (open source). Obviously other solutions exist on the market and the intent of this document is not to exclude any of them.

The architecture shows a potential mix of DTR interfaces based on:

- Information services;
- Web Applications;
- Mobile APP.

The data server would be based on a single solution assuming native replication capabilities of the technology used to store the data.

The technical infrastructure (TI) aspect is based HTTP and the TI essentially would operate an application/web server to perform HTTP based exchanges over TCP-IP.

The main format both for input and output would be GML not excluding more formats to be added/supported.

The picture below shows in green the applications and in orange the information services. It should be noted that an application itself may use information services, hence when under the form of a web application it comes as a preconfigured application. The names of services, interface, applications and operations are fictitious but should be sufficient to understand the overall intent of the architecture:

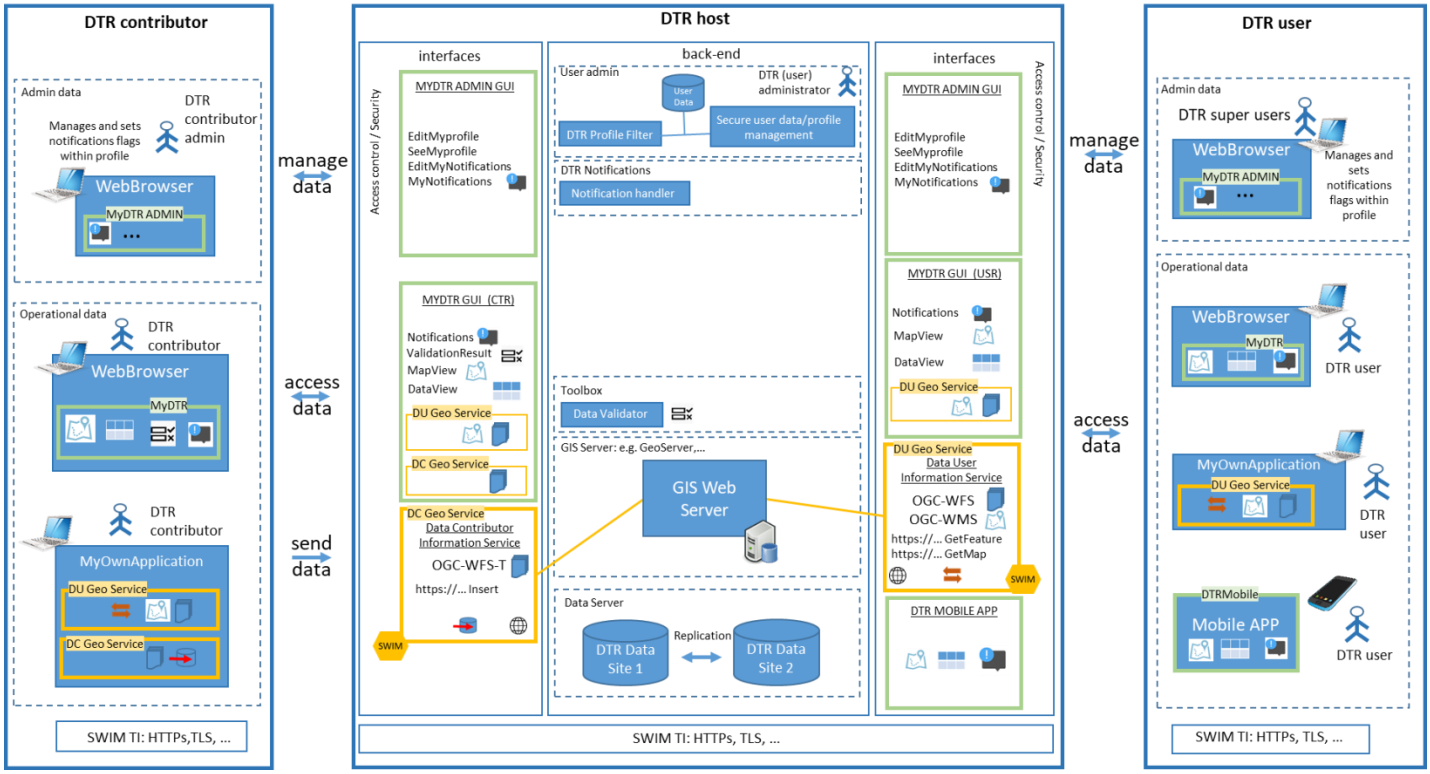

Figure 9: DTR architecture view

Key characteristics:

- database approach augmented with GIS server and OGC interfaces capabilities;
- mature technology available; and
- community of vendors and integrators (OGC community).

7.2.1. Discussion

For the DTR contributor, this architecture view is based on the use of transactional OGC WFS-T (T for transactional) to write data into the DTR store. This would imply a GML writing capability on the contributor side, as GML would be the format used over WFS-T. Whilst GML can become complex to handle the intent is to use a simple feature schema design knowing that the only geometry type to be supported is points and the number of properties is limited, see Appendix B.

The timely handling of data reception, storing and further distribution will require attention for performance reasons. However, if message format payload is kept simple this approach could work. Obviously prototyping would have to demonstrate.

Towards the DTR users, the GIS server application could facilitate multiple format exposure adding GeoJSON, KML, and GeoPackage. These are also the formats supported by OpenLayers [Ref 09].

The GIS web server may come as open source (e.g. GeoServer [Ref 13]), with relevant functionality for the deployment of the DTR, which may in fact come close to resolve many of the needs of the DTR, and facilitate the exposure of information services using OGC WFS and WMS implementation specifications. Related open source components could be beneficial when it comes to developing contributor or user web applications. Typically the use of OpenLayers on the client side lowers the entry level for geospatial visualisation.

Furthermore, at data server level, when based on GeoServer, the typical possible open source database could be PostGIS, an extension of PostGres. Replication strategies can be developed using PostGres. Again, the listing of these solutions does not prevent the use of any other solution available from the market.

From a TI point of view this architecture essentially binds to the TI over HTTPs, using mostly get, and post methods.

The TI requirements associated with supporting this approach can be based on the EUROCONTROL SWIM TI YP specification using the following link from the EUROCONTROL SWIM Service Registry: https://eur-registry.swim.aero/reference/ectl-swim-tiyp-v1-0/requirements

- The mandatory requirements can be obtained by selecting “mandatory” in combination with the WS-light interface binding as filters.

TLS	SWIM-TIYP-0008	The Service Interface Binding shall support the following versions of the Transport Layer Security Protocol (TLS): + IETF RFC 5246 (TLS v1.2)
HTTP	SWIM-TIYP-0009	The Service Interface Binding shall support HTTP/1.1.
HTTP over TLS	SWIM-TIYP-0010	The Service Interface Binding shall comply with IETF RFC 2818 (HTTP over TLS).
TLS Authentication	SWIM-TIYP-0042	The Service Interface Binding shall support one of the following authentication mechanisms for TLS: + Mutual authentication with X.509 certificates + Server authentication with X.509 and Client authentication with HTTP Basic or HTTP Digest.
HTTP Status Code Header	SWIM-TIYP-0043	The Service Interface Binding shall be able to use the HTTP Status-Code header.
HTTP Reason Phrase Header	SWIM-TIYP-0044	The Service Interface Binding shall be able to use the HTTP Reason-Phrase header.

Figure 10: EUROCONTROL SWIM Specification TIYP: WS-light interface binding mandatory requirements

7.2.2. Alternative considerations

7.2.2.1. Database engine with geospatial support

An alternative could be to disregard the dedicated GIS web server and use the geospatial capabilities/extensions offered by database vendors. As an example Oracle Application Server could be used in combination with Oracle spatial extensions including an OGC WFS engine. In this case the TI interface binding can either be SOAP or XML based. The WS-light binding requirements (see above) are applicable when using XML based requests only. Alternatively, should the approach be to use SOAP, the requirements associated with supporting SOAP can be based on the EUROCONTROL SWIM TI YP specification using the following link EUROCONTROL SWIM Service Registry: https://eur-registry.swim.aero/reference/ectl-swim-tiyp-v1-0/requirements The mandatory requirements can be obtained by selecting “mandatory” in combination with a SOAP interface binding of choice.

7.2.2.2. Geospatial support only for DUIS

An alternative could be to consider OGC support towards the DTR user only as an add-on to the architecture focused at global interoperability for mainstream GIS applications. In this case for instance a specific interface on the DCIS side would be defined not using the WFS-T/GML mechanism to write into the DTR store. For instance it could be opted to go for a REST style interface. These aspects relate to making decisions on format. [Discussion topic: see 8.1]

7.2.3. SOAP, WFS, REST

Whilst format considerations are important also interface bindings are to be considered. The following is a high level discussion in order to establish a level playing field taking key technology approaches into consideration: SOAP / OGC-WFS / REST.

Applications use services to communicate. Services have interfaces which need to be accessed. Two essential things happen:

- Interaction with the service through a message;
- Transport of content between consumer and provider through a message.

Interaction is based on standard protocols or conventions such as: SOAP, WFS, and REST.

Transport is based on standard protocols such as HTTP.

SOAP (Simple Object Access Protocol) is a messaging protocol for web services which is being widely used to implement (web) services. SOAP relies on XML but not exclusively over HTTP. The use of SOAP relates to using other W3C Web Service standards.

REST (Representational State Transfer) is a gaining approach to access service interfaces which builds on HTTP methods to form restful URLs which the provider exposes to the consumer. Service replies are not tied to the use of XML. Most commonly replies are encoded in JSON, however not exclusively. But when thinking about REST this frequently equates to the combination of HTTP and JSON whereby interfaces are restful and perceived as a resource on the web. Note that OGC is currently standardizing REST interfaces for WFS. A draft is available at http://docs.opengeospatial.org/DRAFTS/17-069.html.

WFS is also based on the use of HTTP methods. URLs can be built that for instance encode a query using key value pairs. Overall WFS specifies a remote procedure call. WFS has also specified SOAP bindings should the choice be made to use SOAP. The use of WFS implies replies encoded in XML, more particularly GML. Towards, the future, the WFS specification is planning to embrace REST and JSON.

Overall the direct use of HTTP over secure connections is a good way to ensure platform independency. Opting for WFS does not mandatory imply the use of SOAP. GIS application servers that implement WFS (also in combination with other OGC specifications such as WMS) may provide more output formats as part of their capabilities, also including REST interfaces (see also GeoServer REST API).

7.3. Overview

The following picture provides a high level architecture overview of the DTR when for instance based on GIS web server approach:

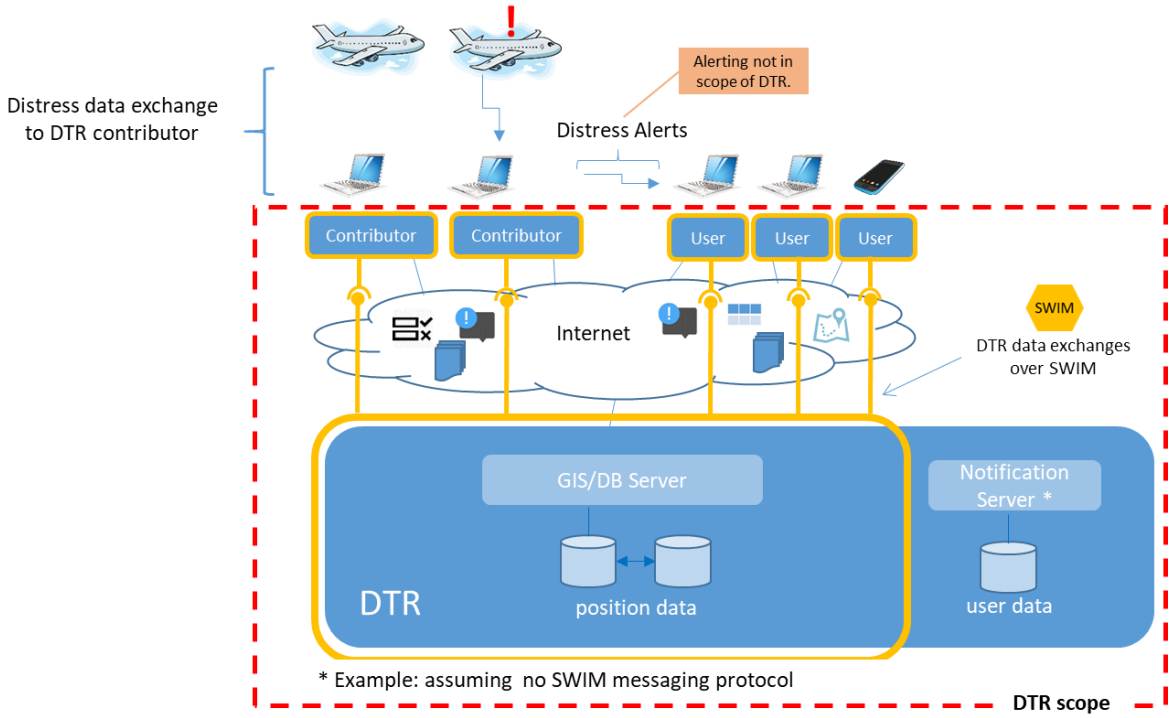

Figure 11: high level architecture overview

Characterizing the architecture based on the approach described:

GIS and database oriented. The entry-point used for the development of the DTR is based on a GIS oriented solution in combination with a database. In terms of complexity induced costs, geospatial enablement appears to be a feasible objective. From the DTR stakeholders and DTR user point of view, global, timely and interoperable usability of the information is probably the primary concern of the DTR. When it comes to global interoperability the OGC geospatial standards are a good choice given the nature of the data to be exchanged. The solution does not support real-time usage of last known position information using for example a messaging middleware.

COTS solution. Solutions available off the shelf may involve less or more architectural components out of the box potentially as open source (e.g. GeoServer):

- GIS application server;
- User data administration;
- Technical infrastructure interface bindings support.

Solution choices probably depend on efforts needed to develop the DTR (customisation, integration), i.e how much of it would be out of the box?

Cloud. The deployment strategy could be more or less involving cloud as opposed to on site hosting. User data administration, GIS application servers, data servers can be obtained in the cloud as services. Some cloud services provide rich platforms including multiple possibilities (e.g. Azure, Amazon, …) Considering cloud aspects, decisions should be taken in how far parts of the DTR solutions could be operated in the cloud, hence not requiring DTR host capabilities and resourcing.

This relates to making deployment choices. These aspects depends on the business model envisaged for the DTR.

Additional capability. The architecture could include more or less additional capabilities (e.g. global FIR data, validation service), offered as additional information services. These capabilities could be added stepwise.

Notification. For notification purposes towards DTR users an e-mail approach could be opted for. The current document does not investigate any messaging middleware to perform this function. This remains an aspect to be investigated by the solution provider. For notification purpose towards DTR contributors it could be considered to handle this at the technical infrastructure level. This remains an aspect to be investigated by the solution provider.

Protocols. The key protocols considered:

- HTTP for web interaction (through secure connection);
- OGC WFS on top of HTTP for geospatial data exchange;
- SMTP if e-mail notifications;
- LDAP for exchange of user profile data.

Gateway aspects to AFTN are considered outside the scope of this document, but the intent is not to exclude any possibility.

Choice of protocol for notification should be consequential to approaches taken.

Test system. The DTR specification identifies the need for a test system. This is an important aspect which will need to be foreseen during the prototyping phase. It will be required to perform contributor validation before accreditation.

8. Discussion topics

The topics below should be discussed for clarification decision purposes.

8.1. Format

Question: what format to use for the DTR?

Given the nature of the data one entry point to the discussion is to separate between formats dedicated for geospatial data ("GIS formats") and those not so.

8.1.1. Dedicated geospatial formats

Some possibilities are listed to contain the set of options available:

DTR input over DCIS:

- GML3.2 is based on XML enabling use of WFS-T out of the box
- GeoJSON is based on JSON requiring REST interface definitions

DTR output of DUIS

- GML3.2
- GeoJSON as possible add-on
- Geopackage as possible add-on
- KML as possible add-on

8.1.2. Non-dedicated geospatial formats

DTR input over DCIS:

- bespoke schema for data input
- FIXM

DTR output of DUIS:

- bespoke schema for data output
- FIXM only if FIXM input retained

8.1.3. Considerations

The scope of the DTR data is geo-temporal. This means that position information and time information is important since an important end-user use case will be to visualize/map the location of the last known position. In all cases the semantics of the information needs to be preserved hence it is important to correctly handle coordinate information in relation to position on earth either by including coordinate reference system information or by clear business rules.

If on the input side of the DTR a non-dedicated geospatial coding is retained the format can be defined such that it facilitates the creation of dedicated geospatial format on the DTR output side. Given the nature of the DTR data the GML representation is however simple. Therefore the difference with a non-dedicated XML based coding is minor, hence using GML would not introduce additional cost but leverage its advantage of good geospatial handling. See also Appendix B.

The Flight Information Exchange Model (FIXM) [Ref 15] captures Flight and Flow information that is globally standardised. For instance, FIXM is being used to define flight information messages in the context of FF/ICE-1. Whilst FIXM is considered non-dedicated to geospatial, it's transformation into a dedicated geospatial format should be feasible (i.e. relatively easy).

If no FIXM is received on the DTR input side it is probably not wise to create FIXM on the output side. The reason for this is that the DTR should not originate flight information. A FIXM input can however be transformed into a dedicated geospatial format such as GML for instance using a transformation service. Opting for FIXM on the input side will obviously require contributors to talk FIXM which for some may be feasible but not necessarily for all. Furthermore a dedicated FIXM schema (as a profile) would need to be built to avoid designing a solution beyond the scope of what is required by the DTR. What is most important with regards to FIXM is that FIXM data can easily be converted into data fit for the DTR and the data used from the DTR can be used a context where the FIXM is used. Hence, the DTR should take into account that upstream and downstream data processing may actually involve FIXM formatted data or services. Therefore the design of the DTR should make FIXM data transformations feasible to the extent possible. This is also why it is important to relate the data format back to the AIRM to ensure common understanding. [See Appendix B]

Should JSON and REST be considered then GeoJSON should be used. Obviously this would imply more bridging work into FIXM environments which are XML based. It is understood though that an approach based on JSON has traction in the industry. It could be an alternative on the input side (DCIS) when not considering the choice of WFS-T/GML3.2.

8.2. Performance measures

Question: what are the peak performance measures which the DTR needs to handle?

DTR data reception: peak loads and performance measures on the DCIS. A one minute rate of 200 byte messages is used to give indication about the needed database capacity. This is defined as a minimum. For more detailed considerations on the architectural choices to make, the maximum rate needs to be identified so to determine the maximum load on the DCIS. This should include an estimate on the maximum number of concurrent active ADTs. Overall this should lead to a maximum time acceptable for the data to travel from the ADT into the DTR.

DTR data distribution: peak loads and performance measures on the DUIS. It is expected that none of the DTR users should wait longer than 30 sec to obtain a last known position requested from the DTR. The time needed to issue a request, for the reply data to travel over the network, i.e. the time after it leaves the DTR host router, and the time taken by DTR user to process the data, is not a DTR host performance responsibility. The measure of a peak consumption load should be established. For example 3 requests per minute for a number of concurrent users hence supporting a maximum data peak throughput of x KB per sec).

Further clarification is required on the overall latency accepted between data origination and data reception by DTR users. It is considered to be the overall time needed for DTR data reception and data distribution.

8.3. Last known position determination

Question: what is the algorithm used by the DTR query handler to determine the last known position from the available information in the DTR store?

There could be the understanding that different contributors may receive/calculate different distress information for the same aircraft.

Subsequently the DTR may receive, for the same distress event, position information from different contributors.

In this case the DTR handling of inputs should be based on a recording approach that adds records to the store as they are received and relates them to the same distress event. Hence an updating approach of the same record is not followed. Every update leads to a new added record which removes the need to perform record locking for transaction purposes.

The last known position is based on the last generated information from the ADT. The DateTime stamp at which this occurs is to be provided. In the AIRM context this is known as the lastRevision information of type DateTime. The DTR query handler uses this lastRevision information to determine the last known position. These are the T1,T1' and T1'' times shown in the figure below.

This is shown in **Figure 12**:

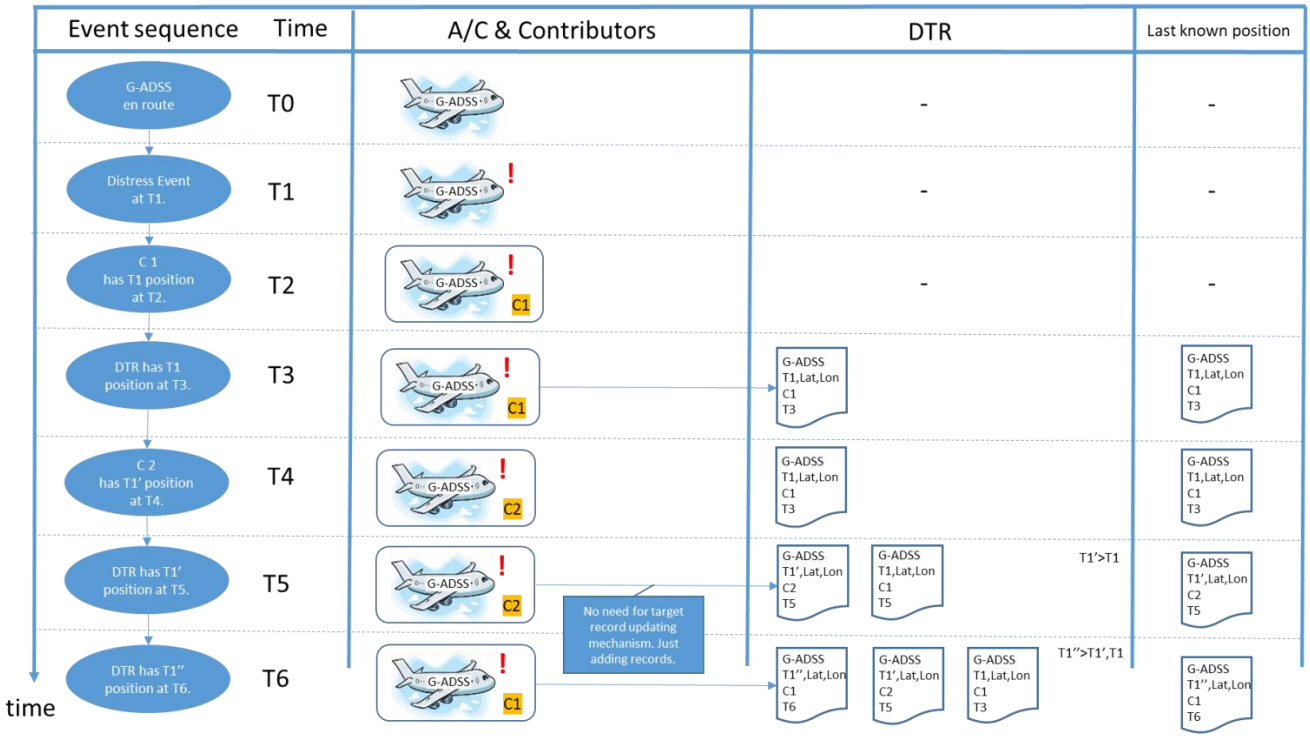

Figure 12: DTR last known position determination

8.4. Notifications

Question: what is the approach regarding notification?

DTR contributor notification: the handling of these notifications at the TI level is a possibility to be investigated. The decision is whether this is an acceptable solution. The advantage is that the notification directly relates to the data messages sent by the contributor.

DTR user notification: the technology needs to be chosen. As a minimum an e-mail approach can be considered but an approach using messaging technology should not be excluded. The solution implementer should investigate and propose.

Reception acknowledgements should be considered since it is good to be aware whether DTR users units actually noted the activity.

Messages should be standardised and computer interpretable to facilitate DTR user application logic supporting the retrieving of information from the DTR.

8.5. Quarantined data

Question: How should the operational requirement of quarantining being translated in an IT concept and/or architectural considerations and/or technical capabilities?

Whilst the DTR sends a notification to the contributor when a message is for example incomplete, a user shall be made aware that the message has be quarantined and is available for consultation.

Storing a message that is corrupted needs more discussion in terms what is exactly required, what is the added value compared to implied cost and complexity of the system as it may imply human intervention. Can't the information not simply be resent by the DTR contributor in case of loss or corruption during transmission?

9. Next steps

This sections elaborates on the to-do's of the DTR development.

The following series of steps are envisaged (non-exhaustive list):

- Step 1: Architecture workshop
  - The participants should come to a more fine grained common understanding on the intended design of the DTR
  - The maximum load on the system needs to be expressed
  - Notification aspects need to be clearly understood and agreed upon
- Step 2: DTR implementation specifications
  - Functional specifications need to be expressed as system requirements that can be used by a solution provider to build a solution fit for purpose.
- Step 3: Elaboration of building blocks
  - Data schemas
  - Information service definitions
- Step 4: Call for interest to build a prototype
  - The demonstration is attributed to an entity
  - The inputs are the specification details and building blocks
- Step 5: Building of prototype and demonstration
  - The activity includes a description of the architecture (deployment diagram) and any recommendations for subsequent deployments and updates to the requirements to be taken in to account.
  - The builder of the prototype should describe the actual architecture proposed. This results in a fine grained definition and allocation of information services to system components (e.g. including process diagrams, sequence diagrams to capture behaviour).
  - This includes the actual actors and their roles.
  - The testing aspect needs to be taken into account
  - Based on the provided "to-be" descriptions this leads to materials necessary to contract and commit on the building and delivery of a DTR system, i.e. the DTR deployment.
- Step 6: DTR Deployment
  - The building of the system is attributed, financed and delivered.
- Step 7: DTR Operations
  - The system is operated and sustained

Obviously the above steps are an example of how the project could be structured and need to be consolidated taking the reality of the "managing entity" that would lead the project into account.

Appendix A – Background on SWIM

As defined in ICAO Doc 10039 SWIM is "*System Wide Information Management (SWIM) consists of standards, infrastructure and governance, enabling the management of ATM related information and its exchange between qualified parties via interoperable services.*" More practically, SWIM is about standard and interoperable interfaces over TCP/IP. SWIM interfaces come in two kinds:

- Information Services: i.e. application level interfaces, and
- Technical Infrastructure interfaces (technology protocols) which in turn are of two kinds:
  - Technical Infrastructure interface bindings (groupings of protocols)
  - Network interface bindings (on top of IP)

The following figure depicts the main components of SWIM along the interoperability stack:

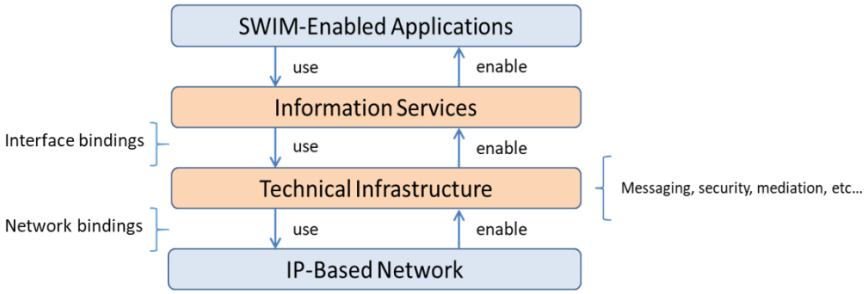

Figure 13: SWIM "stack"

From the above picture, it becomes clear that SWIM introduces a layer of "information services" that use "technical infrastructure" capabilities.

Note: whenever an interface is called programmatically it is understood to be an API assuming a given calling coding environment from where to invoke the call.

Designing a SWIM solution is about devising an interoperable system in a multi-stakeholder and connected environment. The following main aspects emerge:

Connecting applications: One aspect of the design decisions to be taken is in terms of which information services are needed, and how they interact with technical infrastructure elements (see figure above). This resolves the question how applications interact over interfaces with other applications connected to the IP network using technical infrastructure capabilities. The choices lead to the interoperability of the system, i.e. "connected applications".

Distributing capabilities: whilst interoperability is a key SWIM concern, yet functionality in any networked system is distributed over multiple systems and capability areas. When performing together they become "one". Here the decisions and agreements contribute to the topology of the overall system: it answers questions such as "where is which component running?" These decisions should take into account considerations, such as operational aspects, business aspects, legal aspects, economical aspects etc… The distribution of the capabilities (i.e. their allocation to systems) determines the capabilities which the contributing actors need to implement or use as shared resources.

Since SWIM is fundamentally a paradigm for information sharing in a connected world, the notion of information service is key. When building applications that provide/consume SWIM information services they become SWIM enabled applications. The information services are the means to exchange the information hence they also connect to where the information resides (e.g. a database, a queue, a file).

Consequently applications, information services, and information "stores" are distributed over:

- the information providing and consuming actors in SWIM;
- shared resource which the actors in SWIM decide to use together (i.e. systems "in the middle" as is envisaged for the DTR)

The following picture summarizes the playing field in which a SWIM system is designed:

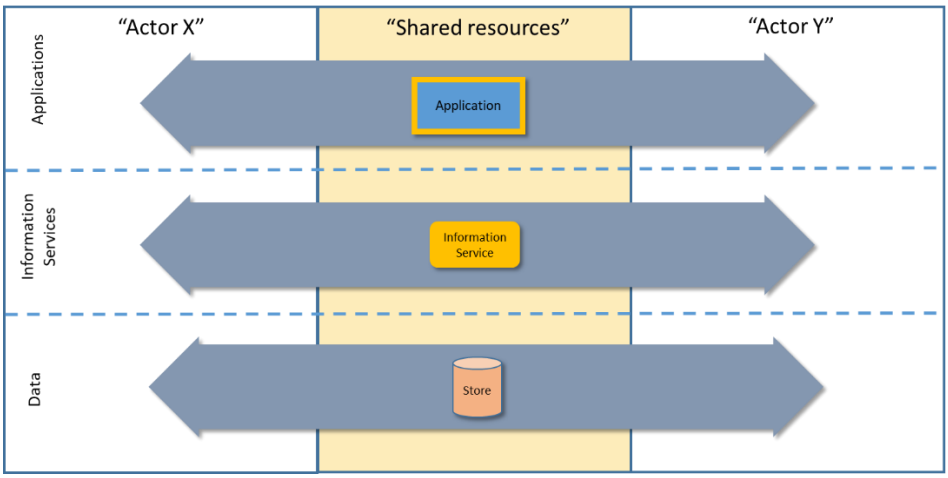

Figure 14: distributing capabilities in SWIM

Interaction between the capabilities:

Based on SWIM, applications, and information services interact hence connecting entities at the business/operational level which requires due consideration of the behaviour of these interactions. Whilst a decoupling on the technical level remains a SWIM goal, there may be time ordered sequences that need to be considered, in particular when data exchanges are event triggered, as is the case in the DTR. This collaborative aspect is at the heart of connecting businesses at the process level. The GADSS operational concept contains some business process diagrams, hence from an architectural point of view sequence diagrams will help to grasp additional behavioural aspects. Furthermore, when referring to interactions and data exchanges these are realised in SWIM by sending/receiving messages which are the result of SWIM service interface operation invocations. When applications connect through SWIM information services they do so through application level messages that are transported over the SWIM technical infrastructure. The following picture shows the high level components that constitute a SWIM service:

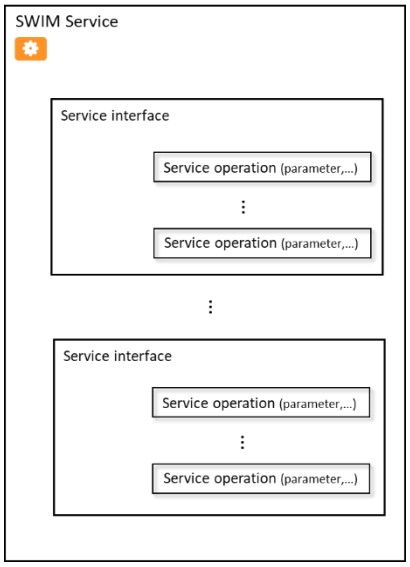

Figure 15: high level components of a SWIM service

Message Exchange Patterns (MEP) and messaging

Data exchanges and notifications are based on message exchanges which follow more or less complex communication MEPs. There is no singular way to realize a publish/subscribe message exchange pattern. In reality the pattern can be realized in various ways. This may or may not involve dedicated system components using messaging technology based on protocols such as AMQP 1.0, SOAP/WS-N, or API's such as JMS.

Based on W3C definitions, a message exchange pattern is defined as: "*A template that describes relationships of multiple messages exchanged between interacting components to accomplish a single complete information exchange*". When a sequence of messages that are exchanged between systems is repeatable then it becomes a pattern which can be used to reason about system interaction behaviour.

MEPs are used at the application level to specify how application level interactions are supposed to be organised when exchanging messages.

Application level MEPs:

- One-way (synonym to "Fire and Forget")
- Synchronous Request/Reply
- Asynchronous Request/Reply
- Publish/Subscribe Push
- Publish/Subscribe Pull
- Brokered Publish/Subscribe Push

Note: Publish/Subscribe is understood to be asynchronous by design.

Application level interfaces, such as DTR information services use lower level protocols as supported by the SWIM Technical Infrastructure. The fundamental patterns used at the lower level are:

- One-way (synonym to "Fire and Forget")

- Synchronous Request/Reply

Using these fundamental patterns any of the (complex) application level patterns can be implemented.

Furthermore, when doing so some fundamental characteristics of the message exchanges emerge (non-exhaustive list):

- cardinality of systems involved in the distribution of information: 1x1, 1xn, nxn;
- mutual knowledge of system addresses to perform exchange: coupled (peer-to-peer), decoupled (via broker);
- mutual system process behaviour to perform exchange: synchronous (processes locked), asynchronous (processes not locked);
- subscription mechanism involved: the consumer expresses interest in obtaining certain messages as offered by the provider i.e. publisher.
    - Publish/Subscribe Push:
        - Subscription: the consumer uses a Request/Reply MEP to subscribe to the publisher.
        - Publication: the publisher uses a One-way MEP to send messages to the consumer.
    - Publish/Subscribe Pull:
        - Subscription: the consumer uses a Request/Reply MEP to subscribe to the publisher.
        - Publication: the consumer uses a Request/Reply MEP to get the messages from the publisher.
        - Notification: the publisher uses a One-way MEP to send a notification message to the subscribed consumer informing on availability of new information.
    - Publish/Subscribe Brokered:
        - Subscription: the consumer uses a Request/Reply MEP to subscribe to the broker who is an intermediate that holds the messages from multiple publishers.
        - Publication: the broker uses a One-way MEP to send messages to the subscribed consumer.
        - This pattern decouples publishers and consumers:
            - Publishers send messages to the broker not knowing to whom they will be forwarded.
            - The moment at which the exchange between broker and consumer happens is independent from the moment that the publisher sent the message to the broker.

Messaging is a technical infrastructure capability. Whereas messaging provides data exchange and distribution capabilities. Messaging can be realized in various ways:

- AMQP 1.0 [Ref 07], messaging may include the concept of queues subscribed to by consumers when instantiating the Broker Publish/Subscribe MEP;
- SOAP/XML WS-N;

- HTTP Get method results in the exchange of a message (i.e. XML file) between the HTTPs client and HTTPs server (i.e. pull);
- HTTP Post method results in the exchange of a message (i.e. XML file) between the HTTPs server and HTTPs client (i.e. push);

Background on SWIM TI related protocols

- OGC WFS for serving geospatial vector format content (request/reply)
  - can be embedded in SOAP
- SOAP/WS-N (publish/subscribe)
  - OGC has published a Publish/Subscribe SOAP binding that is based on the WS-N Based Notification and Brokered Notification [Ref 12]
  - Whilst this is a recently published standard the use of WS-N should be considered carefully since it requires clients to implement the WS-N Consumer role in order to receive messages.
- AMQP 1.0 (publish/subscribe)
  - this is a vendor neutral wire level protocol
  - AMQP 1.0 can support the MEP patterns described above
  - An AMQP network is built up from nodes and links between the nodes of the network. Producers (nodes) generate AMQP messages. Consumers (nodes) process AMQP messages. Producers and consumers exist in client applications. Queues (nodes) store and forward AMQP messages. Queues exist in broker applications. Queues are used when a broker configuration is retained as solution.
  - AMQP 1.0 does not enforce the use of a broker “in the middle”.

Constraining SWIM components

As indicated before, SWIM is based on choices which SWIM actors make together to interoperate, i.e. these choices are formalized in specifications which are agreed to be standards. These specifications constrain the implementation to augment interoperability.

The Eurocontrol specifications address the following concerns through three specifications:

- EUROCONTROL Specification for SWIM – Service Description [Ref 06]
- EUROCONTROL Specification for SWIM – Information Definition [Ref 05]
- EUROCONTROL Specification for SWIM – SWIM Technical Infrastructure Yellow Profile [Ref 04]

The following picture summarizes their high-level purpose:

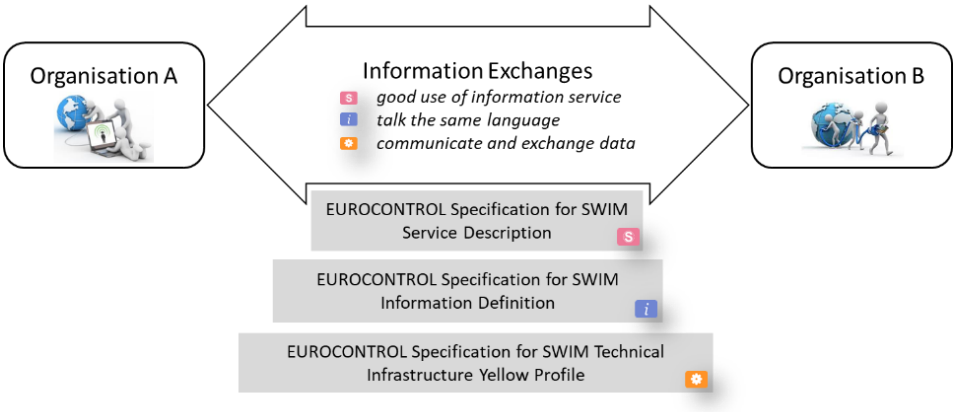

Figure 16: EUROCONTROL SWIM Specifications

Since the DTR aims to be based on SWIM, the Eurocontrol SWIM specifications provide the foundational and constraining requirements which add meaning to achieving a design and state of the DTR system considered as "to be SWIM":

EUROCONTROL Specification for SWIM – Service Description

The service description specification expresses requirements on the information to be provided about SWIM information services. This leads to understanding what the information service does, how it works and how to implement a consuming client. The aim is to describe implemented services in accordance with the EUROCONTROL SWIM service description specification. In addition the specification could also be used to share information about a non-implemented service, i.e. during design time.

EUROCONTROL Specification for SWIM – Information Definition

The information definition specification expresses requirements on semantic interoperability of information exchanged through SWIM information services. This leads to using a common language and eventually to interoperable data based on agreed formats. Whilst the EUROCONTROL specifications do not prescribe format standards, the DTR design will need to make choices, hence the OGC implementation specifications can complement at this level. The aim is to achieve a state wherein the information exchanged over DTR information services are semantically corresponding to the AIRM in accordance with the EUROCONTROL SWIM information definition specification, and in addition to provide specifications for the encodings of the information (formats).

EUROCONTROL Specification for SWIM – Technical Infrastructure Yellow Profile

The aim is to implement according the protocols listed in the EUROCONTROL SWIM technical infrastructure yellow profile specification. The technical infrastructure specification expresses requirements which constrain the choice of the technical protocols to be used.

> Documents can be found here: http://www.eurocontrol.int/publications/eurocontrol-specifications-system-wide-information-management-swim

Appendix B – AIRM mapping and DTR Input format

DTR data to AIRM mapping:

Field	Format	Correspondence with AIRM
Latitude	Double/coordinate	urn:aero:airm:1.0.0:LogicalModel:Abstract:GeoEnabledEntity@position
Longitude	Double/coordinate	urn:aero:airm:1.0.0:LogicalModel:Abstract:GeoEnabledEntity@position
3LD	string	urn:aero:airm:1.0.0:LogicalModel:Subjects:Stakeholders:Stakeholder:AircraftOperator@designatorICAO
Aircraft Registration (with Nationality Mark)	string	urn:aero:airm:1.0.0:LogicalModel:Subjects:Aircraft:Aircraft@aircraftRegistration
Contributor Code	string	urn:aero:airm:1.0.0:LogicalModel:Abstract:TemporalEnabledEntity@source
Altitude (m)	number	urn:aero:ses:eurocontrol:airm:1.0.0:LogicalModel:Subjects:Flight:AircraftAltitude@flightLevel urn:aero:airm:1.0.0:LogicalModel:DataTypes:MeasureTypes:ValDistanceType@uom urn:aero:airm:1.0.0:LogicalModel:DataTypes:UnitsOfMeasure:CodeUomLengthType@M
Altitude (ft)	number	urn:aero:ses:eurocontrol:airm:1.0.0:LogicalModel:Subjects:Flight:AircraftAltitude@flightLevel urn:aero:airm:1.0.0:LogicalModel:DataTypes:MeasureTypes:ValDistanceType@uom urn:aero:airm:1.0.0:LogicalModel:DataTypes:UnitsOfMeasure:CodeUomLengthType@FT
Heading	angle	urn:aero:ses:eurocontrol:airm:1.0.0:LogicalModel:Subjects:Flight:AircraftDirection@heading
Airspeed (km/h)	number	urn:aero:airm:1.0.0:LogicalModel:Subjects:Flight:AircraftSpeed@indicatedAirSpeed urn:aero:airm:1.0.0:LogicalModel:DataTypes:MeasureTypes:ValSpeedType@uom urn:aero:airm:1.0.0:LogicalModel:DataTypes:UnitsOfMeasure:CodeUomVelocityType@KM/H
Airspeed (kt)	number	urn:aero:airm:1.0.0:LogicalModel:Subjects:Flight:AircraftSpeed@indicatedAirSpeed urn:aero:airm:1.0.0:LogicalModel:DataTypes:MeasureTypes:ValSpeedType@uom urn:aero:airm:1.0.0:LogicalModel:DataTypes:UnitsOfMeasure:CodeUomVelocityType@KT
Date and time of transmission	DateTime	ISO DateTime
Date and time of receipt	DateTime	ISO DateTime
Position Timestamp	DateTime	urn:aero:airm:1.0.0:LogicalModel:Abstract:TemporalEnabledEntity@lastRevision

Note: the date and times are combined in one ISO DateTime type. Transmission and Reception DateTime are outside the scope of the AIRM since more related to the messaging aspects a the TI level

GML snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<dtr:AircraftState xsi:schemaLocation="www.airm.aero.dtr dtr_gml.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:dtr="www.airm.aero.dtr">
    <dtr:aircraftRegistration>BALOST</dtr:aircraftRegistration>
    <dtr:designatorICAO>ERT</dtr:designatorICAO>
    <dtr:flightLevel uom="FT">23</dtr:flightLevel>
    <dtr:flightLevel uom="M">23</dtr:flightLevel>
    <dtr:heading>270</dtr:heading>
    <dtr:identifier>1234</dtr:identifier>
    <dtr:indicatedAirSpeed uom="KT">160</dtr:indicatedAirSpeed>
    <dtr:lastRevision>2019-05-02T15:55:00Z</dtr:lastRevision>
  - <dtr:position>
      - <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
            <gml:pos>3.14159265358979 3.14159265358979</gml:pos>
        </gml:Point>
    </dtr:position>
    <dtr:source>TrustMe@data.com</dtr:source>
</dtr:AircraftState>
```

XML snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<dtr:AircraftState xsi:schemaLocation="www.airm.aero.dtr dtr_pox.XSD"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dtr="www.airm.aero.dtr">
    <dtr:aircraftRegistration>BALOST</dtr:aircraftRegistration>
    <dtr:designatorICAO>ERT</dtr:designatorICAO>
    <dtr:flightLevel uom="FT">23</dtr:flightLevel>
    <dtr:flightLevel uom="M">23</dtr:flightLevel>
    <dtr:heading>270</dtr:heading>
    <dtr:identifier>1234</dtr:identifier>
    <dtr:indicatedAirSpeed uom="KT">160</dtr:indicatedAirSpeed>
    <dtr:lastRevision>2019-05-02T15:55:00Z</dtr:lastRevision>
        <!-- Position is "Latitude followed by longitude".-->
    <dtr:position>3.14159265358979 3.14159265358979</dtr:position>
    <dtr:source>TrustMe@data.com</dtr:source>
</dtr:AircraftState>
```

Notes:

- Above examples are purely illustrative mostly aimed at demonstrating the minor difference in terms of complexity between both options. The GML example uses “urn:ogc:def:crs:EPSG::4326” to refer the coordinate reference system (i.e. WGS84) explicitly.
- The key discriminator to determine the last known position available in the DTR is the data and time the position data was originated. This information is captured in the lastRevision element in the examples above.

Appendix C – Multiple DTR considerations

Based on the DTR specification the DTR comes as a single, i.e. a “one-stop-shop” for DTR contributors and users. This Appendix contains considerations in the event that multiple DTRs would be built to support a global deployment.

In this case the introduction of the “DTR Region” notion should be considered to indicate the geographic region covered by each DTR. Each DTR Region would then replicate the DTR architecture in terms of the standardised interfaces. This further leads to the following, assuming that contributors and users are only linked to one DTR, even though a contributor/user may operate in more than one DTR Region:

- Each provider can choose to connect to their preference regional DTR.
- Each user can choose to connect to their preference regional DTR.
- When a DTR contributor provides data to the DTR of choice (e.g. DTR 1 for Contributor A in the figure below), the interested user may actually be connected to the DTR of another DTR Region (e.g. User G connected to DTR 2 in the figure below). This then could lead to the following possible mitigations:
    - Through message forwarding: Each DTR becomes itself a contributor to the other DTRs. Each DTR forwards messages to all other DTRs.
    - Through data harvesting: the DTRs harvest from each other at database level, and all DTR have all DTR data (e.g. using a data replication mechanism). In this case the obligation of each DTR to replicate data (within a region) may actually be reconsidered since duplication would anyway happen.
- The overall DTR administrator needs to ensure that the union of all DTR Regions covers all the geographical space covered by international air transport.

The figure below illustrates the described scenario(s):

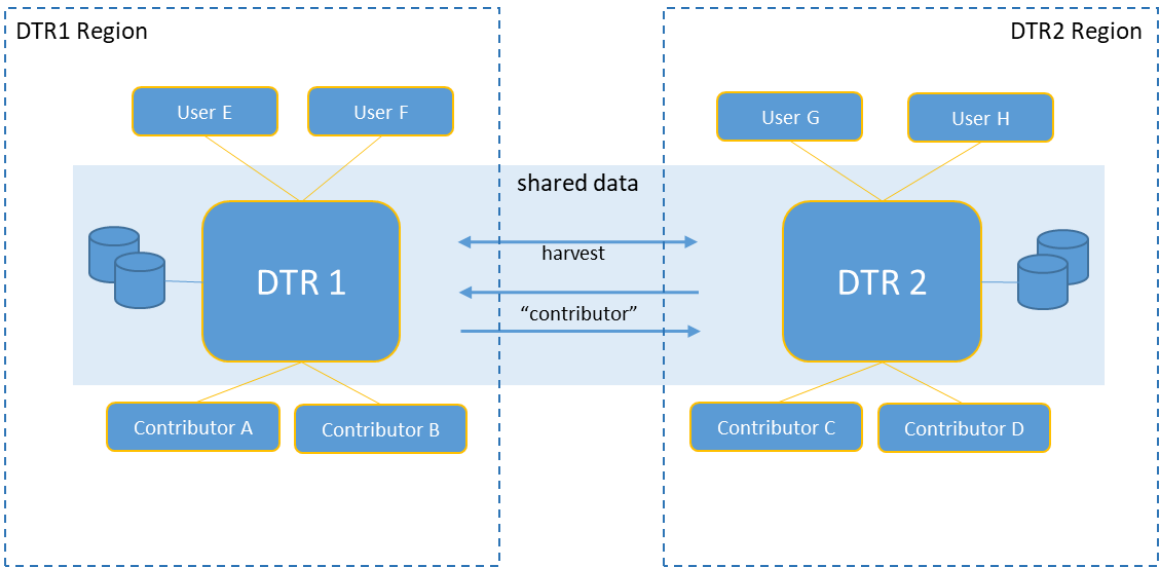

Figure 17: Multiple DTRs