Requirements for Identity Resolution and Enrichment Service

written by Marton Papp

01. September 2015

Requirements Version: Commit [None] VCS Id [topics]

Contents

Chapter 1

IDRES Requirements

Requirements of the Identity Resolution and Enrichment Service

1.1 Identity Resolution and Enrichment Service

Description: The *IDRES* will be implemented as a central component that provides vessel identity resolution and enrichment services to its service consumers.
Rationale: The *IDRES* is a service that provides information about vessels and their identities to other system components. It implements nontrivial procedures to maintain a non-amiguous mapping between identifiers that are used to refer to vessels and the physical identity of vessels. It is a central repository of vessel related information that maritime applications will refer to, in order to provide a consistent picture of vessel identity and vessel details to the maritime users.
Solved by: 1.2.1 Implemented as a Service, 1.3.1 Mapping Vessels to Business Identifiers, 1.5.1 Updating the Database, 1.4.1 Vessel Details

Id: Idres

1.2 Architecture

This section contains requirements about the general architecture of *IDRES*

1.2.1 Implemented as a Service

Description: *IDRES* **must** be implemented as a central service that is accessed by its service consumers using a well defined API.
Rationale: The API must be designed to be as stable as possible so that future changes in the service implementation will not break the functionality of existing service consumers. The API should have the characteristics of a request response or an asynchrunous message driven interface. Its usage needs to be clear and well documented and must not require implementing any complex logic on the service consumer side.
Depends on: 1.1 Identity Resolution and Enrichment Service
Solved by: 1.2.2 Extensibility, 1.2.3 High Availability, 1.2.4 Scalability

Id: Arch0Service

1.2.2 Extensibility

Description: *IDRES* **must** be implemented with extensibility in mind, so that in the future it can be adapted to new requirements or to changes in the existing requirements with relatively low effort.
Rationale: *IDRES* will eventually be used by several applications for varios purposes. The functionalities that it provides are expected to grow in the future. So is the number of data sources that it will possibly be connected with. It is not possible to foresee all the possible future use cases and requirements that will need to be implemented. Therefore many of its functionalities need to be designed to be extensible and prepared for future development.
Note: This is a general requirement for the entire architecture of the service. The individual and necessary extension points will be identified by their own requirements in this document.
Depends on: 1.2.1 Implemented as a Service

Id: ArchExtensibility

1.2.3 High Availability

Description: *IDRES* **must** be implemented with a focus on minimizing the duration when a planned intervention or an unexpected incident causes the service to be unoperational and therefore impact the functionality of its service consumers.
Rationale: *IDRES* will be a central component that many other applications will rely on for their proper functioning. Failure in the central component can have a significant impact on the entire infrastructure. Hence the service must be designed to be tolarent to faults of hardware and software components and designed to be responsive in most conditions.
Depends on: 1.2.1 Implemented as a Service

Id: ArchHighAvailability

1.2.4 Scalability

Description: *IDRES* **must** implemented with scalability in mind, so that the number of requests that the service is able to respond to without any major degradation in the performance is proportional to the hardware and software resources that are allocated to it.
Rationale: The frequency of the requests that the service will need to respond to depends on several factors like the number of service consumer applications, the number of reporting vessels, the reporting frequency of individual vessels, the area covered by message receivers, etc. These factors vary between reporting systems and in general they are expected to increase over time without any well defined upper limits. For this reason *IDRES* must be be able to handle increased load assuming that the necessary hardware resources are available.
Depends on: 1.2.1 Implemented as a Service

Id: ArchScalability

1.3 Identity Resolution

This section contains the requirements of the functionality of identifying the sending vessels of messages that are received in a message report stream.

1.3.1 Mapping Vessels to Business Identifiers

Description: *IDRES* **must** maintain a mapping between logical vessel identifiers that are referring to existing physical vessels and business identifers that are used in the maritime world to identify vessels taking into consideration the dynamic nature of business identifiers.
Rationale: Business identifiers in the maritime world are assigned by different organization in different countries over different continents. Business identifiers might change for a given physical vessel and the same business identifer that was once assigned to a certain vessel might be reassigned to a different vessel in the future. Errors might be introduced in the assignment of business identifiers that might take some time to be resolved by the assigning authority. *IDRES* must be prepared to handle such and similar cases and maintain a consistent database of identifiers that will be consulted and regarded as a central source of truth by the service consumers.
Depends on: 1.1 Identity Resolution and Enrichment Service
Solved by: 1.3.2 Vessel Identifier, 1.3.3 Resolving Identity of Senders in a Message Stream, 1.3.4 Resolving Logical Identifier from Business Identifer and Timestamp, 1.3.5 Resolving the Business Identifiers from Logical Identifer and Time Period, 1.3.6 Extensible Message Stream Support, 1.3.7 AIS Message Stream Support, 1.3.8 LRIT Message Stream Support, 1.3.9 VMS Message Stream Support, 1.3.10 Time Dimension in Mapping Database, 1.3.11 Unambiguous Identity Resolution

Id: Map0BusinessIdentifier

1.3.2 Vessel Identifier

Description: *IDRES* **must** introduce the concept of a logical vessel identifier that uniquely identifies a single physical vessel that exists or existed during a certain period of time.
Rationale: None of the existing business identifiers in the maritime world is suitable to uniquely identify all tracked vessels. Therefore a new internal identifier needs to be created for the purpose, the values of which are unique and provide a clear one-to-one mapping between physical vessels and identifier values.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapLogicalIdentifier

1.3.3 Resolving Identity of Senders in a Message Stream

Description: *IDRES* **must** provide a service for its consumers that resolves the identity of senders of messages in a continuous message stream taking into consideration the related performance requirements.
Rationale: The frequency of messages in the incoming streams depends on several factors like the number of reporting vessels, the reporting frequency of individual vessels, the coverage of the message receivers, etc. These factors vary between reporting systems and in general there are no well defined upper limits. Hence the identify resolution should be optimized for performance in order to be able to keep up with the message rate of the stream.
Note: Message reports are normally sent by a specific onboard device at a certain rate. The message contains certain business identifiers that are assigned to the vessel. These identifiers can be used to correlate subsequent messages from the same vessel. The business identifiers change infrequently if at all during the lifetime of the vessel. The total number of tracked vessels is expected to grow slowly over time. Hence the size of the database that contains all the business

identifiers for all the tracked vessels together with their history is relatively small and slowly extending. It is expected to be feasible to replicate this entire dataset within a few seconds and keep it synchronized with the central source over the network with a minimal resource requirements. This could be one possible way to achieve the required performance of performing identity resolution at the rate of the message stream.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers
Solved by: 1.3.4 Resolving Logical Identifier from Business Identifer and Timestamp

Id: MapMessageStreamResolution

1.3.4 Resolving Logical Identifier from Business Identifer and Timestamp

Description: *IDRES* **must** provide a service that takes a certain business identifer and a given timestamp as the input and responds with the logical identifier of the vessel that the given business identifier was assigned to at the given point in time. Similar service must be provided for all supported business identifers.
Rationale: The reponse to these requests will contain the logical identifer of the vessel that is or was trasmitting the given business identifer at the given point of time. These requests will be used by applications that are processing message streams or historical messages in a certain area. The implmentation must be optimized for performance as the requests can be frequent depending on the volume of messages delivered by the stream or the activity of users accessing historical data.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers, 1.3.3 Resolving Identity of Senders in a Message Stream

Id: MapReqBusinessTimestampToPeriod

1.3.5 Resolving the Business Identifiers from Logical Identifer and Time Period

Description: *IDRES* **must** implement an endpoint that provides all the business identifiers that were transmitted by a vessel with the given logical identifer over a given period of time.
Rationale: These queries will be used used by applications that need to reconstruct the track of poisitions of a vessel that is identified by its logical identifer. The service must provid a corresponding query of this kind for all the supported business identifer domains. As some business identifers of a single physical vessel might change over time, the response must contain a all the business identifers that were transmitted by the vessel during the given time period.
Note: Since the business identifiers are changing infrequently the response will contain no more than one identifier in most of the cases. Still, it is possible that a request will be made over a time period during which the requested vessel did change its business identifier. In order to correctly handle this case the API must be defined in a way so that it can respond with more than on identifer together with the corresponding time period.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapReqLogicalPeriodToListofBusinessPeriod

1.3.6 Extensible Message Stream Support

Description: *IDRES* **must** provide support for developing modules that support messages streams that are not specifically defined in this document.
Rationale: It is likely that the service will need to process further message streams in the future apart from the ones that are specifically mentioned in this document. Theferore the implemention must provide extension points to add support for new message sources. The development of such modules must be supported by documentation and examples.
Note: Ideally the support for message streams that are specifically mentioned in their own requirements should all be built upon a common base component that provides generic support for stream based data. This way the particular implementations could serve as a reference for developing further modules of the same kind.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapStreamExtensibility

1.3.7 AIS Message Stream Support

Description: *IDRES* **must** support identity resolution for senders of messages in an AIS stream
Rationale: The AIS system defines several different types of messages that are transmitted by vessels and coastal stations. The primary business identifier found in the AIS messages is the MMSI number. The MMSI number is a 9-digit decimal number. The first three digits correspond to the flag state of the vessel. The MMSI number can change over the lifetime of the vessel, for example when the vessel changes flag state. MMSI numbers can be reused, meaning that an MMSI that was once assigned to a certain vessel might be assigned to a different one at a later point in time.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapStreamTypeAis

1.3.8 LRIT Message Stream Support

Description: *IDRES* **must** support the resolution of the identity of the sender of messages in the LRIT messages stream.
Rationale: Normally LRIT messages are sent once per 6 hours and contain the IMO number as the primary identifier of the vessel. The IMO number is a numberic value (fits a 32 bit integer) and is assigned to the vessel for its entire lifetime. IMO numbers are not reused.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapStreamTypeLrit

1.3.9 VMS Message Stream Support

Description: *IDRES* **must** support the identity resolution of senders of VMS messages
Rationale: VMS messages are sent by certain fishing vessels at a frequency of about 1 message per one or two hours. For European vessels the primary business identifer in VMS messages is the IR number. The IR number is a sequence of 12 alphanumeric characters, the first three being letters and the last 9 being digits. For the vessels that do not have an IR number the primary busiess identifer to be used in a VMS message is the radio call sign. The radio call sign is a sequence of alphanumeric characters of a maximum lenght of 7 and it can change over the

lifetime of a vessel. Radio call sign can be reassigned to different vessels when they are no longer used.
Note: In practice the IR nubmers do not always respect the standar pattern. It might be safer to expect 12 alphanumeric characters without any further restrictions. It is not yet known if the IR number can change or be reused. For now it is safest to assume that it can.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapStreamTypeVms

1.3.10 Time Dimension in Mapping Database

Description: *IDRES* **must** provide services that resolve the identity of a vessel in the context of a certain point or period of time.
Rationale: Business identifiers can change during the lifecycle of a vessel. The other way is also possible, that is, the same business identifier can possibly be assigned to different vessels during different periods of time. Some service consumers will want to resolve the identity of the sender of messages that arrived at some point in the past. In other cases the service consumer will want to know the business identifiers that were assigned to a certain physical vessel over a particular period of time. In order to support these requirements the *IDRES* must store in its database not only the latest state of mapping between business and logical identifiers but also the history as it has changed over time. The requests and responses that the service supports must also reflect this by including the dimension of time.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapTimeDimension

1.3.11 Unambiguous Identity Resolution

Description: *IDRES* **must** provide services that give unambiguous responses to vessel identity resolution requests.
Rationale: In many scenarios the service consumers will not have the opportunity to perform any manual resolution or even automated procedures that take considerably long time. This is the case for example when a component is processing a stream with a high message rate. In these cases that consumer will require that the service provides the identitiy of a certain vessel on a best effort basis. It might well be possible that a later request to the service with the same input leads to a different result, for example because in the meanwhile the service had received further information and updated its database accordingly. This is because in some cases the information that is necessary to correctly identify the sender of a certain message will reach the *IDRES* later than the message itself is first processed. The service consumers must be prepared to handle such cases.
Depends on: 1.3.1 Mapping Vessels to Business Identifiers

Id: MapUnambiguity

1.4 Vessel Details

This section contains requirements about the functionality of providing details about the attributes, status and events of vessels.

1.4.1 Vessel Details

Description: *IDRES* **must** support the storage and retreival of certain attributes that are attributed to physical vessels and assigned to their logical identities in the database.
Rationale: Apart from the business identifiers it is often necessary to display or process certain information about vessels that is related to their physical attributes or actual status. This information should be stored centrally and made available to other applications so that they can present a consistent view about the vessels to the outside world. In many ways the processing rules that apply to business identifiers also apply to all the vessel details. The main difference is that the vessel details are not used in the process of identity resolution and their values are normally not required to be unique in the maritime domain. There are different kind of possible values that are to be stored as vessel details and they are described in their corresponding requirement specifications.
Depends on: 1.1 Identity Resolution and Enrichment Service
Solved by: 1.4.2 Dynamic Vessel Details, 1.4.3 Enrichment of Messages with Vessel Details, 1.4.4 Vessel Events, 1.4.5 Vessel Details Extensibility, 1.4.6 Vessel Details Requests, 1.4.7 Static Vessel Details

Id: VD0VesselDetails

1.4.2 Dynamic Vessel Details

Description: *IDRES* **must** provide support for storing and retrieving vessel details that are dynamic in the sense that their value can change during the lifecycle of the vessel.
Rationale: Some vessel details, like the name of the vessel can change during the lifetime of the vessel. The history of the changes of these vessel details might be interesting for the business, for example in the case when a user is looking for information about the vessel but only knows a former name of it. Some details might reflect some state of the vessel, for example its navigation status, which might change rather frequently. The implementation should be prepared to support values that are expected to change wihout any know limitation on the number of changes that can happen during the lifetime of the vessel. The service needs to be able to store and retrieve the entire history of changes assuming that the necessary hardware resources are available.
Note: In certain use cases, taking the enrichment of near real time position messages as an example, the history of dynamic values might not be important, only their most recent state. The implementation should support the calculation and retrieval of such a projected value for dynamic vessel details in an efficient way.
Depends on: 1.4.1 Vessel Details

Id: VDDynamic

1.4.3 Enrichment of Messages with Vessel Details

Description: *IDRES* **must** support the scenario of extending certain messages with some information about the vessel details of the sender of the message.
Rationale: Some existing applications use message streams as their only input source for their processing logic. In order to satisfy their information requirements some additional data about the sender is added to each message in the stream before they reach the processing application. In the future it is expected that these applications will be redesigned to use a central service to get that information, which is a more scalable approach. However, to support the current architecture, it is currently necessary to continue enriching certain message streams with vessel

details. The performance requirements of this functionality is similar to that of the vessel identity resolution service for message streams.
Note: It can be assumed that the enrichment will only be used for message streams that are processed in real time. Therefore the enrichment will only require access to the latest state of the vessel details but not their history. Similarly to the business identifier this might be done based on a set of data that is limited in size, growth and update frequency, which makes it possible to achieve a highly performant implementation by using local replication.
Depends on: 1.4.1 Vessel Details

Id: VDEnrichment

1.4.4 Vessel Events

Description: *IDRES* **must** support storing certain events that are related to vessels, happen at a speicific time during the lifetime of a vessel and can be described using a set of arbitrary attributes.
Rationale: Vessel events and dynamic vessel details are closely related in the sense that changes in vessel details are normally the consequence of certain events in the lifetime of the vessel. Very often one can be deducted from the other. The decision whether a certain information should be modeled one way or the other or maybe both ways depends on the requirements of how the information will need to be accessed later. In fact most of the elements of message streams, like position reports, status reports, voyage reports, vessel notifications can be modeled as events and these events can be processed to maintiain a set of dynamic vessel details that can be queried in an efficient way. Similarly to dynamic vessel details the implementation should be prepared to handle a history of events that is only limited by the amount hardware resources that is allocated for the purpose.
Depends on: 1.4.1 Vessel Details

Id: VDEvents

1.4.5 Vessel Details Extensibility

Description: *IDRES* **must** be implemented in a way so that the actual set of vessel details that are strored and made accessible by the service can be changed following well defined and documented procedures with a relatvily low effort.
Rationale: The information that is needed to be maintained about vessels and the way that this information needs to be accessed will change over time. In order to be able to adapt to future changes the schema of the stored data and the associated update procedures and retrieval services should be separated from the main application logic and should be able to be changed wihout having a global impact.
Note: Changing the schema of the vessel details does not need to be done dynamically. It is acceptable to shut down the service and maybe some service consumers for doing this kind of update.
Depends on: 1.4.1 Vessel Details

Id: VDExtensibility

1.4.6 Vessel Details Requests

Description: *IDRES* **must** implement a sophisticated interface for querying the vessel details database that takes into consideration the dynamic nature of the database, the performance requirements of the consuming applications and the different type of values that the vessel details database is required handle.
Rationale: Different applications will want to query the vessel details database in various ways. The type of queries will be similar to those that one would expect in any current database system with the additional particularities of the data that is extended with the dimension of time. The type of requests include: searching for a particular value in the static or dynamic vessel details, restricting the search for a dynamic vessel detail for a particular time or period, retrieveing a list of changes for particular dynamic vessel detail for a given time or period. For events it is necessary to list a certain type of events of a particular vessel for a given time period, or find the closest event of a certain type before or after a certain point in time.
Note: The actual requests do not need to be able to be updated at runtime. Shutting down the service in order to do the update is acceptable.
Depends on: 1.4.1 Vessel Details

Id: VDRequest

1.4.7 Static Vessel Details

Description: *IDRES* **must** support storing vessel details that are static by nature, that is, they do not change during the lifecycle of the vessel.
Rationale: Some physical characteristics, like vessel length or date of construction do not change during the lifetime of a vessel. There might be other cases, where a certain characteristic might change, but only its latest values is ever relevant for the business. These cases might also be modeled as static values.
Note: In theory, static vessel details could also be stored as dynamic ones for which only a single values is stored for all the history of the vessel. However, static values might give room for optimalization and help to achieve higher performance where it is critical.
Depends on: 1.4.1 Vessel Details

Id: VDStatic

1.5 Updating the Database

This section contains requirements about updates to the database of the service

1.5.1 Updating the Database

Description: *IDRES* **must** provide facilities to continuously update its database by using the information that is retrieved from external data sources.
Rationale: The database of the service needs to be continuously updated in order to correctly reflect the actual attributes and status of the vesseld that it is modelling. The updated state must be reflected in the responses that are sent to service consumers in near real time.
Depends on: 1.1 Identity Resolution and Enrichment Service
Solved by: 1.5.2 Dynamic Update Logic, 1.5.3 Manual Conflict Resolution, 1.5.4 Update Database from Message Stream

Id: Upd0Update

1.5.2 Dynamic Update Logic

Description: *IDRES* **must** implement a mechanism that makes it possible that the logic that performs the updating of the database of the service is defined and changed dynamically during runtime without any downtime that is noticable by the service consumers.
Rationale: The logic that updates the identity mapping and vessel detils database is the most complex element of the service. Experience shows that it is very difficult to describe a processing mechanism that works correctly for all the possible input streams for an extended period of time. Hence it is necessary to retain the possibility of redefining the processing rules in order to improve the quality of the resulting database without any impact or intervention in the hosting environment and consuming applications.
Note: The processing logic could possibly be implemented by using some scripting language that can be replaced during runtime and interpreted by the processor that is responsible for updating the database. Alternatively it could be implemented using a compiled language and ensuring that the newly compiled binaries can replace the old ones during runtime. Ideally the implementation would support both solutions or leave room for future alternative solutions. Note that the eventual implementation will need to consider the high performance requirements that apply to high rate message streams.
Depends on: 1.5.1 Updating the Database

Id: UpdLogicDynamic

1.5.3 Manual Conflict Resolution

Description: *IDRES* **must** provide an interface for notifying an operator when some data arriving in the input streams could not be automatically merged into the database according to the actual update logic. A conventient graphical user interface must also be provided in order to analyse the conflict and provide a manual resolution.
Rationale: There is very low control over the quality of some data streams that are used to update the database of the service. It is likely that there will be occurences of conflicts of data that the actual update processing logic will not be prepared for. Theses situations will need to be resolved manually by an operator. During this process statistics will be collected about the conflicting information that will later help to improve the processing logic.
Depends on: 1.5.1 Updating the Database

Id: UpdManualResolution

1.5.4 Update Database from Message Stream

Description: *IDRES* **must** be able to process a stream of messages and extract any information that is relevant to maintain the vessel identity and vessel details database.
Rationale: Most of the information that is required to maintain the database of the service is available as a message stream. Even those sources that are not availabe as a stream can be converted to a stream by custom adapters and processors. Being able to use streams as the input to the servcie will therefore cover most of the different types of data sources.
Note: In case when the database of the service needs to be synchronized with an external database that is not available as a stream of changes an external process can be implemented that regularly converts the entire external database to stream of messages and sends it to the service for processing.
Depends on: 1.5.1 Updating the Database

Id: UpdStream