Integrasjonsrammeverk

Dokumentadministrator: Frode Junge	Gyldig fra: 13.12.2019	Revisjon: 1.0
Godkjent av: Bjørn Våga	Revisjonsfrist: 12.12.2020	ID: 2945

Integration Framework

Content

1 Introduction

1.1 Change History

Version	Changes	Date	Author
0.8	First draft	25.01.2019	Rune Hollås/HPAS
0.9	Tilpasset HMN	02.12.2019	Stig H. Solum og Frode Jünge
1.0	Godkjenning av ledergruppe POD	10.12.2019	Frode Jünge

1.2 Purpose

The purpose of this integration framework is to give guidelines for integrations within HMN and between HMN and surrounding applications. These guidelines must be used when planning and documenting the integration and will contribute to choose the most suitable integration mechanism.

To support HMN's objectives it is necessary to have a high level of ambition for application integration. This will contribute to information sharing and effective work processes.

Applications can be linked either at the back-end (data exchange) or the front-end (client integration).

1.3 Outside scope

This document does not include guidelines and patterns for development on the ESB itself.
Laboratory device and medical device integrations are not covered by this framework.

2 HMN's integration strategy

2.1 National Architecture principles

The HMN integration strategy shall be compliant with the national architecture principles applicable to the health sector issued by National ICT [1]. The table below gives an overview of the architecture principles and the impact the principle has on integration strategy in HMN.

Principle	Description	Impact on integrations

Principle	Description	Impact on integrations
Holistic approach	A holistic approach is to be used when assessing needs, changes, opportunities and solutions. This involves taking into account the overall usefulness for the health service in the Central Norway Health Region.	When designing an integration it shall be assessed if there are other needs that can be covered by the integration or if an existing integration can be expanded. Aim at flexibility, modularity and simple management
Process orientation	The enterprises will through process orientation (e.g., patient flow and other core processes, as well as supporting processes) realise consistent and coherent health services, and ensure that ICT-solutions are designed to support these processes.	An integration shall use the workflow or activities the integration shall support as a starting point, and aim at simplifying and/or automating workflow or collaboration between actors (systems or persons)
Service orientation	Service orientation shall be a basic design principle for enterprises and their ICT-solutions. This applies to all domains of enterprise architecture (business, information, application, and technology).	The integration architecture shall support loosely coupled applications. Integration services shall be described in a service catalogue to stimulate reuse and avoid duplication of information and solutions.
Interoperability	The enterprises and their ICT-solutions shall be designed for interoperability at organisational, semantic and technical levels.	Open, international (communication and content) standards shall be preferred when establishing integrations, in order to reduce the complexity and vendor dependency. Mandatory national standards shall be followed for national services. Established industry standards or "best practice" shall be used when no international or national standard exist. Proprietary integration interfaces shall be avoided
Information security	The enterprises shall ensure the quality, confidentiality, integrity, availability, and traceability of information.	All integrations shall follow the established security policy. As a rule of thumb, the integration activity shall be logged to ensure integrity and traceability. This includes who has sent what to whom, especially when crossing organisational borders. Exceptions: Registry lookup, master data synchronization
Availability	All relevant user groups shall have access to necessary functionality and information, in the right form at the right time and in the right place.	Integrations shall be robust and be able to provide the required information at any time. The management of integrations shall follow the same rules/SLA as the service/process it supports.
Quality of use	The ICT-solutions of the enterprises shall be designed in a way that ensures effectiveness, efficiency and a good user experience.	A good user experience shall be emphasised when designing integrations, both for end users and administrators of the integration. *Predictability and stability for end users and systems shall be emphasised. Administration and exception handling shall be easily understandable and unified, as much as possible.*
Adaptability	The enterprises' way of organisation, their processes, ICT-solutions, information, and technology shall be designed in such a way that they can support changes, and not act as restrictions for change.	Reusability and use of standardized frameworks, profiles interfaces and information carriers shall be emphasised when designing an integration. Hardcoded parameters and business rules shall be avoided. Integrations shall be scalable in terms of number of users or information rate.

Principle	Description	Impact on integrations
Information management	Information is a critical resource for the enterprises and shall be managed accordingly.	Integrations shall deliver correct information, including information about errors. Integrations shall as a rule of thumb not compensate for errors or shortcomings in the involved applications. Business logic shall preferably be implemented in the application requiring the logic, not in the integration middleware

Tabell 1 Architecture Principles

2.2 Integration architecture

The integration strategy emphasise all IT systems to be loosely coupled by using an enterprise service bus (ESB).

The ESB is a strategic component in the integration between various systems and environments, and provides integration architecture benefits like reusability, encapsulation and distribution.

The ESB will as a rule of thumb handle the technical interfaces, protocols and transform to relevant interchange/content standards when necessary.

International interchange standards are preferred. National standards are preferred where there exist no international standards, typically areas specific to Norway. Proprietary interchange interfaces shall be avoided.

In some cases there might be a need for direct integrations where the solutions handle the interchange standard natively, for instance:

1. When the content has to be signed using personal certificates (e.g. sick-leave note to NAV)
2. Some medical devices and possibly also personal connected health care (PCHC)
3. Special purpose integrations for information critical in its nature (real-time or near real-time requirements)

Some of these direct integrations can be implemented using built-in functions in the solutions. It may however be advantageous to simplify direct integrations by collectively handle (some of) them through third party products, for example a product to handle medical devices.

Figure 1 below shows a conceptual model of HMN’s ESB and how it interacts with the different actors in Central Norway Health Region.

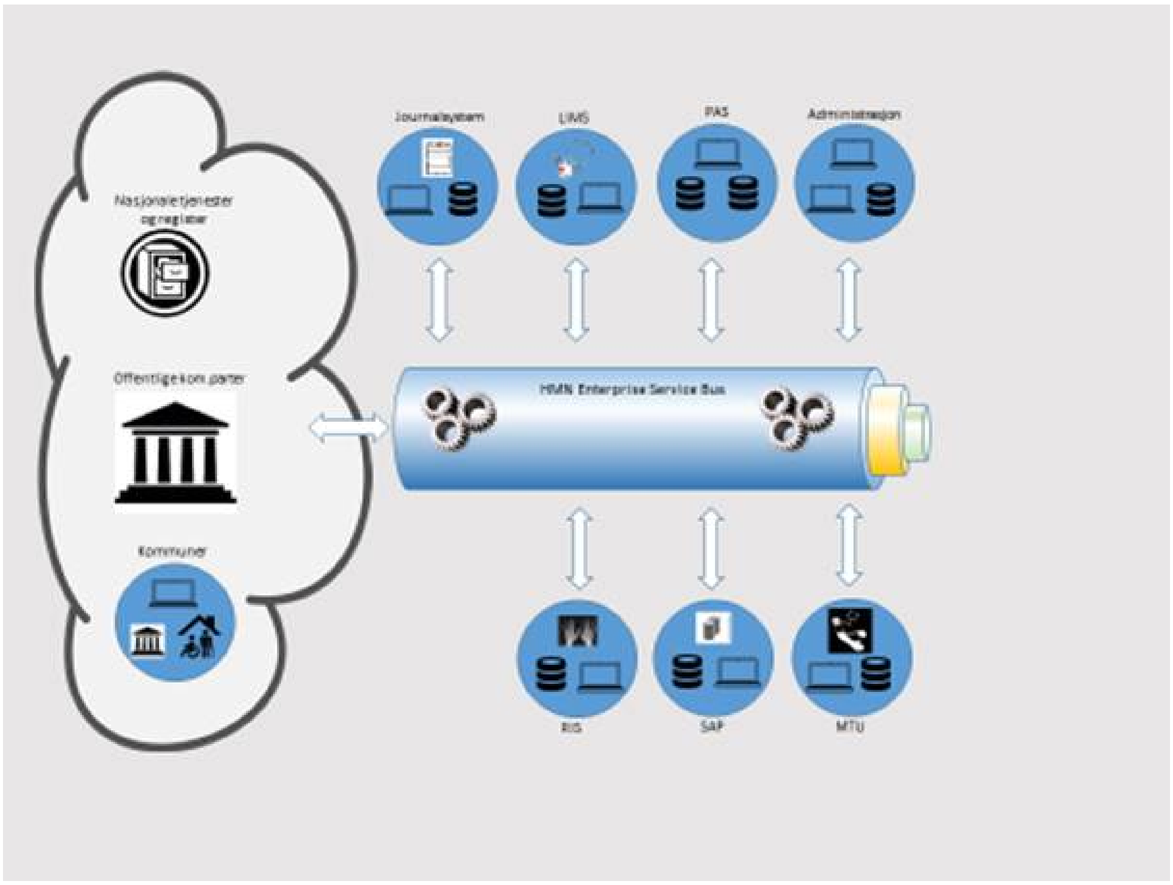

Figure 1 Enterprise Service Bus architecture

In addition to the system-to-system integrations described above, client-to-client integration may be an alternative in some cases. It is a goal to use common mechanisms and international standards (i.e. CCOW) to implement these integrations.

3 Integration principles

The following chapter describes integration principles derived from the architecture principles and the integration architecture described in the chapters above.

3.1 External information exchange

Term	**Meaning**
Principle	All external information exchange shall pass through the service bus
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to avoid having integrations that are directly dependant on other systems. This allows adapting to changes in HMN and/or external systems without having to make changes in both systems, as well as reuse of the integration across multiple communication parties. By passing all information through an ESB, improved control on the exchanged information is obtained. Relevant architecture principles, cf. chapter 2.1: • Service orientation • Adaptability
Consequences *(How to implement and follow – what does it mean)*	All incoming and outgoing messages between HMN and external systems shall pass through HMN ESB, as an intermediate component. Required interface changes shall be managed by the ESB, rather than both end-point systems.

Excpetions *(prefefined exceptions)*	• Medical device integration • Laboratory device integration

3.2 Standardized messages

Term	Meaning
Principle	Integrations shall use standardized communication profiles and content standards.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to have a system which is simpler to integrate with other systems. The use of standardize messages simplifies integration with systems which are changed/implemented in the future. This also allows for using common integration services for several separate systems, which use the same message formats, e.g., patient demographics look-up. Relevant architecture principles, cf. chapter 2.1: • Service orientation • Interoperability • Adaptability
Consequences *(How to implement and follow – what does it mean)*	All integrations shall use international communication profiles and content standards, when possible. When establishing or maintaining integrations, proprietary interfaces or local variants shall be avoided, if there are suitable international communication profiles and content standards for the integration.
Excpetions *(prefefined exceptions)*	• Integrations with mandatory national standards

3.3 Transport guarantee

Term	Meaning
Principle	The ESB shall ensure transfer of data from one system to another, based on a predefined route.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to achieve fast, efficient and secure transfer of data between systems. The service bus shall guarantee that received messages are delivered to the recipient system, or in the opposite to guarantee that sent messages are delivered to the senders receiving system. If the recipient is unable to receive the message, the service bus shall notify the sender. Relevant architecture principles, cf. chapter 2.1: • Quality of use
Consequences *(How to implement and follow – what does it mean)*	Received messages may be schema-validated, but no validation is performed on the content. If the data is not possible to interpret, or the recipient is unreachable, a notification shall be communicated to the sender. If the sender and/or receiver requires it, the service bus shall guarantee the order of the messages.
Excpetions *(prefefined exceptions)*	

3.4 No duplicate checking on the service bus,

Term	Meaning

Principle	The service bus shall not perform any form of duplicate check on the data.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to delimit the responsibility between the service bus and the communicating system. All messages are managed as unique messages by the service bus. The service bus assumes that the sender and receiver takes responsibility for keeping track of which messages are sent and received. Relevant architecture principles, cf. chapter 2.1: • Information management
Consequences *(How to implement and follow – what does it mean)*	Messages shall be managed as unique messages regardless of whether or not they are duplicates. The business logic for duplicate management shall be the responsibility of the sending and/or receiving system.
Excpetions *(prefefined exceptions)*	

3.5 Traceability

Term	**Meaning**
Principle	All information exchange shall be logged in a manner which ensures traceability.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to ensure information integrity and traceability. Relevant architecture principles, cf. chapter 2.1: Information security
Consequences *(How to implement and follow – what does it mean)*	The receipt if a message by the service bus, as well as the actions taken by the service bus towards the message, shall be logged. It shall be possible to track several events occurring with the same information exchange throughout the entire service bus. As a minimum, the following actions shall be logged; received by the service bus, change/transformation, and delivery to the receiving system. The message's unique ID shall follow the message throughout the entire service bus, in order to ensure correct logging. Key information of the message shall be extracted and made searchable by the logging/tracking system.
Excpetions *(prefefined exceptions)*	

3.6 Communication security

Term	**Meaning**
Principle	Consumers outside the secure zone shall not initiate traffic directly to the secure zone.

Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to ensure the security of the secure zone. All data transfer in to or out from the secure zone shall be initiated from within the secure zone. Relevant architecture principles, cf. chapter 2.1: • Information security References: • Code of conduct
Consequences *(How to implement and follow – what does it mean)*	The sender shall transmit the message to a service, which the service bus in turn listens to. The service bus shall initiate fetching of the messages from the service. A reverse proxy (e.g., BigIP) will be the mechanism/service responsible for receiving the incoming request, terminating the request and forwarding it to the service bus.
Excpetions *(prefefined exceptions)*	

3.7 Transformation

Term	**Meaning**
Principle	Transformation from the sender's message format to the receiver's message format shall be done on the service bus, preferably on the BizTalk server.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to consolidate where transformation is performed and maintain flexibility and adaptability to changes in integrations and/or systems. This entails that the receiver system will not have to manage different data formatting from different senders, and reduce the impact of changes in data format in the sending system for the receiver system. Relevant architecture principles, cf. chapter 2.1: • Adaptability • Service orientation
Consequences *(How to implement and follow – what does it mean)*	The messaging contract between the sender and the service bus, and between the service bus and receiver system will identify which formats are valid for the given systems. During transformation of the message, all fields will be transformed to the message format used by the receiver system. The content standard directs the frmat. Changes to field format (typically date format, number format etc.) shall be done on the service bus, so that the different systems won't have to make adaptations to another system's field format in their message format. However, it is important that numerical values are not rounded.
Excpetions *(prefefined exceptions)*	Personally signed content (PKI) cannot be transformed on the service bus.

3.8 Transformation when more than one integration platform is involved

Term	**Meaning**
Principle	Avoid unnecessary intermediate transformations.

Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to avoid having transformation distributed unnecessarily across several intermediate steps along the communication chain, when there are more than one integration platforms involved in an integration. Relevant architecture principles, cf. chapter 2.1: • Quality of use
Consequences *(How to implement and follow – what does it mean)*	Transformation shall be done on one of the integration platforms, and the other(s) should preferably act as a relay. Where transformation should be performed must be agreed upon prior to establishing the integration.
Excpetions *(prefefined exceptions)*	

3.9 Encryption of content

Term	Meaning
Principle	Content 'sent to' or 'received' from systems residing outside the secure zone (HMN in this context), containing sensitive information, shall be encrypted.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to maintain the required security in order to ensure the privacy of the information subjects. Relevant architecture principles, cf. chapter 2.1: • Information security References: • Code of Conduct, factsheet #24
Consequences *(How to implement and follow – what does it mean)*	The contract between the systems outside the secure zone and the service bus shall include a description of how the messages shall be encrypted and decrypted in the message exchange. The service bus shall manage the encryption and decryption of the messages.
Excpetions *(prefefined exceptions)*	

3.9.1 Signing

Term	Meaning
Principle	Messages 'sent to' or 'received' from systems residing outside the secure zone shall be signed with the sender's enterprise certificate
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to enable the receiver of a message to verify the identity of the sender. If required, a personal certificate is used as well. Relevant architecture principles, cf. chapter 2.1: • Information security
Consequences *(How to implement and follow – what does it mean)*	The service bus shall manage validation of certificates on incoming messages, and be able to sign outgoing messages with the sender's enterprise certificate.
Excpetions *(prefefined exceptions)*	

3.9.2 Authentication

Term	Meaning
Principle	All integrations shall conform to authentication mechanisms with security levels appropriate for the information exchanged.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to maintain the confidentiality of information, by ensuring unequivocal identification of the communication parties. Relevant architecture principles, cf. chapter 2.1: • Information security References: • Code of Concuct, factsheet 14?
Consequences *(How to implement and follow – what does it mean)*	Authentication mechanism at appropriate security levels shall be used with all integrations to identify the communication parties. Specifically, this means using PKI at levels 3/Person-Standard or 4/Person-Høyt.
Excpetions *(prefefined exceptions)*	

3.9.3 Enrichment

Term	Meaning
Principle	The service bus shall as an exception enrich a message transmitted from one system to another.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to avoid unnecessary manipulation of content. The content transmitted through the integration should, as much as possible, contain the same information as when it is transmitted out of the source system. Integrations shall as a rule of thumb not compensate for the shortcomings of the involved applications. Relevant architecture principles, cf. chapter 2.1: • Information management
Consequences *(How to implement and follow – what does it mean)*	Integrations shall be implemented with minimal use of enrichment, as far as possible other methods shall be explored.
Excpetions *(prefefined exceptions)*	One exception could be if the sender system and receiver system use different codes/identifiers for master data, e.g., requisitioner identifier. In that case the service bus is responsible for look-up to the correct register, in order to transmit the correct identifier.

3.9.4 Transport receipt

Term	Meaning
Principle	The service bus shall be able to create and transmit a transport receipt (ack/nack), and receive incoming transport receipts.

Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to ensure verification of successful data transfer. Relevant architecture principles, cf. chapter 2.1: • Information security • Information management • Quality of use Reference:
Consequences *(How to implement and follow – what does it mean)*	If required by the sender, the service bus shall when receiving a message offer a transport receipt confirming that the message is received. If the recipient transmits a transport receipt, the service bus shall be able to receive it. This applies to service calls and message exchange with systems/applications. A prerequisite for synchronous transport receipts is the use of adapters supporting synchronous communication. The sender shall be able to wait for a transport receipt in the same transaction as the transmission to the service bus, and the service bus shall be able to receive a transport receipt from the recipient in the same transaction as the transmission.
Excpetions *(prefefined exceptions)*	

3.9.5 Application receipt

Term	**Meaning**
Principle	The service bus shall receive and transmit an application receipt.
Rationale *(Why we have this principle, reference to arch prinsciple or norm)*	The purpose of the principle is to ensure that the data transfer is accepted and understood by the receiving system, or get feedback on status or errors that has occurred in the processing of the message. Relevant architecture principles, cf. chapter 2.1: • Information security • Information management • Quality of use References: Norwegian e-health framework for electronic message exchange in healthcare [2]
Consequences *(How to implement and follow – what does it mean)*	If required the service bus shall be able to receive an application receipt from the receiver system, and transmit this to the sender system. The application receipt confirms that the message is read by the receiver application, and can contain additional information such as status and error messages. This receipt shall be written to the logging system on the ID of the originating message (i.e., the message of which the application receipt is a confirmation). This can for example be the ID of a lab order sent to the LIMS. When required, application receipt is sent as a separate transaction, but requires that the receipt contains a reference to the originating message (unique ID).
Excpetions *(prefefined exceptions)*	

4 Integration specific parameters and attributes

4.1 Integration type

There are two main types of integration that is **Data Exchange** or **Client to client**.
Data exchange is the most common, and commonly described as “Integration”.

Client to client – The main purpose of integrations of this type is to launch an external system from one client system to another on the same client (e.g. PC). Information may be passed as part of the “launch-command”. In some cases this information may be meta information such as user-id and patient-id, or even some more information to pre-populate the screen on the launched system.
Data Exchange - The main purpose of this integration type is **information exchange** between systems, often mentioned as Application-to-application (A2A) or Business to business (B2B). Sometimes data is exported or imported more or less manually from the applications. From an end to end perspective this is still considered to be an integration.

Chapter 4.1 describe patterns and scenarios for data exchange integrations

4.2 Data exchanges and exchange patterns

There are two main types of **data exchanges,** Synchronous and Asynchrounous.
These two has different subsets of (**message) exchange patterns** which is described below.
The description below is based on the following article: *«*
3 Message Exchange Patterns in Application Integration You Should Know About (with Examples)»
[3] Content from the article is cited in *Italic* text.

In general, message exchange patterns that enable data exchange between applications are either synchronous or asynchronous, though a combination of these two are also possible.

4.2.1 Synchronous

“Synchronous communication is also called blocking communication, because all operations in an application that sends a request are blocked until it receives a reply. The connection between the sender and replier (in the API context, this would be typically HTTP connection) stays open during this period of time. This type of communication is essential when the sender application needs an immediate response to continue with data processing.
Synchronous message exchange patterns also mean that all operations are performed one after the other, or in other words, in sequence. This is why using an integration middleware that ensures very high performance with low latency is key in such integration scenarios – after all, data needs to be processed at very high speed so that no business operations get impeded.”

The following exchange pattern(s) are Synchronous:

4.2.1.1 Request-Response

“This is a very typical message exchange pattern of synchronous communication, and it is what it says: An application sends a request to another application, or if an integration middleware is implemented, to the middleware. It then waits for a response, or a timeout.

Example: In a real business situation this would be the case when a support employee needs to call a customer from the interface of the corporate communication tool.

In this scenario, the communication tool doesn’t store customer’s account data, but gets it from a CRM system it is connected to. When the employee clicks on the “Call” button, the communication tool first sends a request for the telephone number to the CRM system, and can perform the “Call” operation only after it receives this number back.”

This exchange pattern shall be used when one is dependent of an immediate answer to be able to proceed.
Typically when two applications is involved in a process cooperation and it is not possible or suitable to continue the process in one application before the answer from the other application arrives.

Examples:
1. Patient demographics request from an external system to the EHR
2. Income request from EHR to NAV.

4.2.2 Asynchronous

"Another name for asynchronous communication is non-blocking communication, because, as you might well have already assumed, the application that sends a request doesn't have to wait for a reply in order to continue operating. The connection between the sender and replier will be closed as soon as the request has been sent out. This also means that multiple processes can occur in parallel.

This type of communication is especially effective when there are large volumes of data that needs to be processed, and is a better fit when no immediate response is expected or required.

Asynchronous message exchange patterns are considered to be more reliable than synchronous, as no application would have a timeout because of waiting for responses, which logically leads to higher services availability. Also, additional functionality can be implemented in the messaging system, and not on the communication ends. But really, it completely depends on the use case. There are integration scenarios when asynchronous communication simply won't work, however fine it is. This needs to be taken into consideration when deciding in favour or against specific message exchange patterns."

The following exchange patterns are Asynchrounous:

4.2.2.1 Asynchronous Request-Response

"Although the Request-Response pattern is actually considered synchronous by nature, there is its asynchronous variation, which is called Request-Callback. In this case, the sender application doesn't have to wait for a response to continue operating. Instead, it sets up a callback process to handle a reply.

Typically, this type of communication would require a specific ID assigned to an original request, as well as a callback address.

Example: The Request-Callback pattern is useful when more than one operation need to be performed in sequence, for example, when an application not only loads data from other sources, but also applies a complex process for its analysis, in which the output of one task is the input of the next one."

This exchange pattern shall be used when one is dependent of a **confirmation**, but can receive the **answer at a later point in time.**
When two applications is involved in a process cooperation or information completion but the triggering application can proceed the process without answer from the
other application.

Examples:
1. Norwegian e-health dialogue message
2. Laboratory order to an external laboratory system
3. Incoming radiology order from external EHR

4.2.2.2 Fire and Forget

"This message exchange pattern is also called one-way, because an application sends a request and continues operating without waiting for a response from receiving application or system, although it will usually expect to get some acknowledgement of that (e.g. a response via webhook). It is a typical pattern of asynchronous communication.

Example: A very classic example would be regular synchronization of data between, say, a cloud-based CRM application and an on-premise ERP system, so that data in both applications is up-to-date. Imaging a Sales employee adding some new account data or changing existing data in the CRM application.

CRM would spot the changes and push them, e.g. into a queue on an integration middleware, from where it will be at some point picked up by or pushed to the ERP system. It doesn't really matter either for the user, or for his/her business processes when the update occurs – immediately or in an hour -, as long as it occurs eventually."

Examples:
1. Batch-transfer of information at a given time, e.g, medication stock

4.2.2.3 Message Routing

"Message Routing is an example of the Fire and Forget pattern when there are more than two applications that need to be integrated.
Example: This might be the case when a company uses one system for billing customers, but two or more CRM systems for various groups of customers, divided by e.g. their locations (South America, North America, APAC) or industries (Automotive, Banking, Pharmaceuticals, etc.).

In this case, an integration middleware is basically a must. Naturally, certain routing rules, and if necessary, even splitting rules (in case only parts of one and the same document will be pushed to appropriate CRM systems) need to be defined properly. But without an integration layer between all applications involved, such an integration scenario will be extremely difficult to address."

Examples:
1. Daily settlement and posting basis to finance systems, when the message shall be routed by the service bus based on the ID of the provider.

4.2.2.4 Publish and Subscribe

"In a way, this message exchange pattern is similar to the Message Routing pattern, only it works the other way round. The receiver applications define what type of data they are interested in. When a sender application pushes data to a so-called 'broker', – in other words, publishes it,- the broker distributes this data in accordance with the receiver applications' specifications.

It doesn't mean, by the way, that there can be only one sender application; like it is with receiver applications, there can be any number of applications that send their data to the broker.

As you might have already guessed, the Publish and Subscribe pattern belongs to the group of asynchronous message exchange patterns.

Example: A good example of this pattern would be an on-premise ERP suite of, say, an automotive manufacturer that collects data from all kinds of sources. This data is needed by multiple various applications, like a CRM, several analytics systems, a billing system, and so on. The advantage of this pattern is that first, it is the most loosely coupled one, – the applications involved doesn't have to know anything at all about each other. And secondly, it presents an ideal way of distributing large amounts of data between multiple applications and systems in a timely manner."

Examples:
1. Death notification message to external systems (e.g. ERP)
2. Information of planned appointment to external systems

4.3 Communication profile

In some cases the external systems facilitates communication and integration using communication profiles. If an integration is realized by using a communication profile, defining which messages to send in which direction and the content standard to use. Communication profiles also includes patterns for receipt-messages to be used

The most common communication profiles used are:
- **ebXml** – used for e-messaging based on the Norwegian e-health/KiTH-standard

- **IHE** (Intregrating the Health Enterprise) – is an initiative by healthcare professionals and industry to improve the way computer systems in healthcare share information. IHE has published a number of profiles for different use-cases and domains within Healthcare. Most known is the Infrastructure profiles (e.g. IHE-XDS).

4.3.1 When to use which profile

If a communication profile should be used, and which one depends on the requirements of external system, communication partner or national regulations. The use of ebXML for national message exchange is mandatory, except AMQP.

4.4 Content standards

There are several content standards in use, the most common are:

- HL7 v2.x, HL7 FHIR, HL7v3, HL7 CDA, ASTM, DICOM, e-helse/KiTH

4.4.1 When to use which standard

Based on the integration principle "Standardized messages" international content standard shall be preferred when possible.

4.5 Technical protocols

There are several Technical communication protocols, the most common are:

- Http, SOAP, REST, POP/SMTP, FTP/sFTP/FTPS, File, MLLP, Queue (MSMQ, AMQP etc)

4.5.1 When to use which protocol

If it is possible to choose the technical protocol to use it shall match the "Data exchanges and exchange patterns". Protocols of synchronous nature shall be preferred when the data exchange is synchronous (e.g. Http, SOAP, REST). For asynchronous data exchanges protocols that are asynchronous shall be used. (e.g. MSMQ, File).
The choice of protocol should also be made based on the "Integration principles".

Sometimes the communication profile or content standard dictates which technical protocol to use.

4.6 Data categories

There are two main categories of data exchanged in integrations:

- Production data, e.g. orders, discharge summary
- Master data, e.g, citizen demographics and health personnel information

5 How to document integrations

This chapter describes how integrations shall be documented.

There are several work products used when describing integrations, see table below

Work product	Purpose	Template owner
Interface blueprint	Figure of all integrations in/out from HMN	HMN
Integration detailed design (IDD)	One document for each integration, with end to end focus (guidance for completion covered by the framework)	HMN
Business requirement document	One document for each integration, with end to end focus (guidance for completion covered by the framework).	HMN
Interface Field-by-field mapping	Document covering field by field mapping for use in transformation	HMN

5.1 How to model integrations

Hopex shall be used for diagrams describing the integration. How to use Hopex is described in "Veileder for bruk av Mega".

5.1.1 Integration blueprint

The purpose of the integration blueprint is a high level visual overview of the integrations from/to a system, with one specific system in focus.
In Mega this may be described in Application Environment diagram

The blueprint shows the integrations, content and endpoints.

The blueprint does not show:

- Middleware/Integration platform
- Route
- Flow/logic
- Tranformation

5.1.2 Integration process

The purpose of integration processes is to associate an integration with the performed processes. At lower levels (i.e., sub-processes), this allows for description of flow/logic, routes and transformation, as well as the distribution of tasks between the different components (i.e., middleware/integration platform and end-points). Where applicable, common processes should be reused across multiple integrations. Typically described as a BPMN-process with swim lanes for the involved components

The integration process does not show:

- Overview of exchanges/exchange contracts between components

5.1.3 Integration sequence

The purpose of the integration sequence is to provide an overview of the dialogue sequence within one single exchange. It may contain simple logic, but mainly describes the order and direction of the flow between two systems. This is described using an Exchange Diagram (BPMN).

The integration sequence does not show:

- Route
- Transformation
- Middleware/Integration platform

[1] https://kilden.sykehusene.no/download/attachments/5177381/20180625_Architecture%20Principles_Version_2.1.pdf

[2] https://ehelse.no/standarder-kodeverk-og-referansekatalog/standarder-og-stottedokumenter-for-elektronisk-samhandling

[3] Olga Haneko 14/10/2016 «3 Message Exchange Patterns in Application Integration You Should Know About (with Examples)» https://www.elastic.io/message-exchange-patterns-application-integration/