TECHNICAL SPECIFICATION

ISO/TS 19468

Second edition
2022-02

Intelligent transport systems — Data interfaces between centres for transport information and control systems — Platform-independent model specifications for data exchange protocols for transport information and control systems

Systèmes de transport intelligents — Interface de données entre centres pour les systèmes de commande et d'information des transports — Spécification du modèle indépendant de plateforme pour les protocoles d'échange de données pour les systèmes de commande et d'information des transports

Reference number
ISO/TS 19468:2022(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*, in collaboration with the European Committee for Standardization (CEN) Technical Committee CEN/TC 278, *Intelligent transport systems,* in accordance with the Agreement on technical cooperation between ISO and CEN (Vienna Agreement).

This second edition cancels and replaces the first edition (ISO/TS 19468:2019), which has been technically revised.

The main changes are as follows:

— UML Communication diagrams have been improved (introduction of Agents and Interfaces to define actor subsystems interactions and addition of subclause 5.8 to describe UML modeling methodology adopted);

— Void input parameter has been defined;

— Description of FEP+EP implementation has been improved with appropriate normative language;

— Publish Subscrube Exchange Pattern has been removed;

— Collaborative Intellignet Transport Systems (ITS) services requirements and features have been reviewed and added in Clause 5 and Annexes B and E;

— Collaborative ITS service FEP+EP PIM description has been introduced in Clauses 10 and 11;

— Annex C has been reviewed, introducing new classes and attributes to support the implementation of features used in other exchange patterns;

— Annex H has been deleted;

— Certain figures have been improved.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document defines a common set of data exchange specifications to support the vision of a seamless interoperable exchange of traffic and travel information across boundaries, including national, urban, interurban, road administrations, infrastructure providers and service providers. Standardization in this context is a vital constituent to ensure interoperability, reduction of risk, reduction of the cost base, promotion of open marketplaces and many social, economic and community benefits to be gained from more informed travellers, network managers and transport operators.

Especially in Europe, delivering transport policy in line with the White Paper[13] issued by the European Commission requires co-ordination of traffic management and development of seamless pan European services. With the aim of supporting sustainable mobility in Europe, the European Commission has been supporting the development of information exchange mainly between the actors of the road traffic management domain for a number of years.

This document supports a methodology that is extensible.

To be able to successfully connect systems and start exchanging data in an interoperable and easy way, there is a need to describe and agree on how this exchange ought to be achieved. This is set out in a data exchange specification. Data exchanges in different scenarios can have different needs and requirements. Therefore, several data exchange specifications can be needed.

Data exchange specifications need to address two main issues. Firstly, they model the stakeholders and actors involved in data exchange, each potentially in different roles, as well as abstract exchange patterns for their interactions. Secondly, they select a suitable implementation platform and clearly specify how the abstract scenarios and patterns are effectively implemented on this platform.

The diagram in Figure 1 shows such an abstract communication scenario from the perspective of a road operator who requires data exchange interfaces between the different components of its own operational systems, either between centre-side components or between centre and field devices, but also to exchange information with other road operators or service providers.

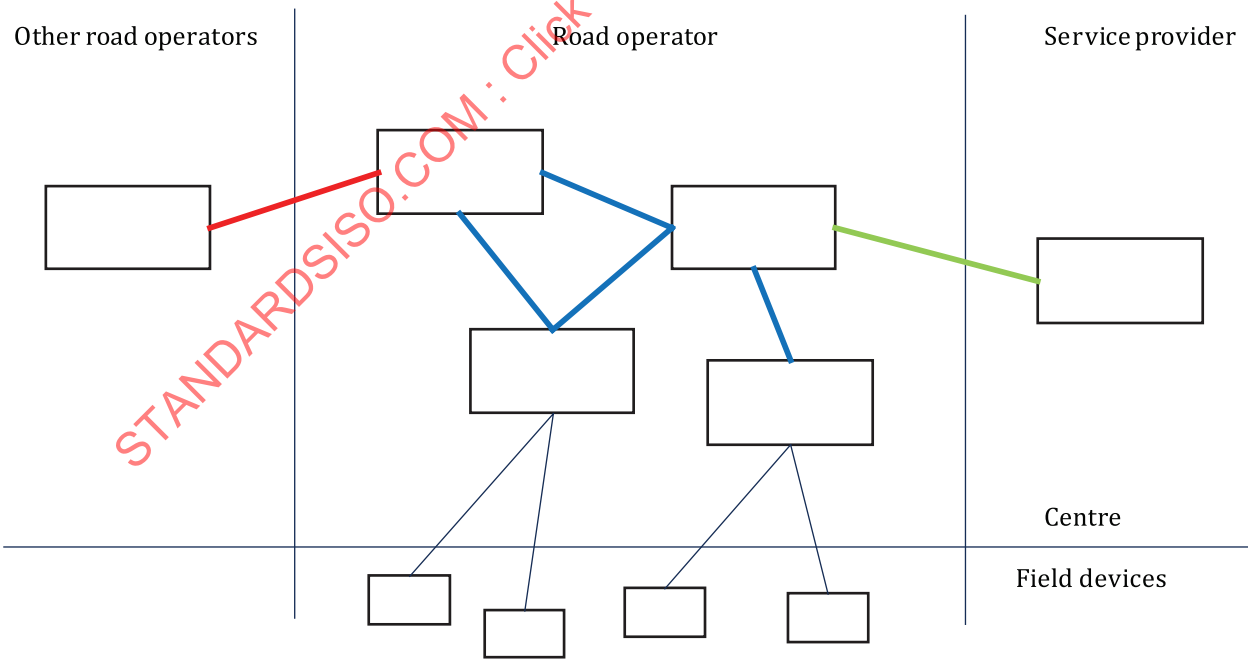

Figure 1 — Abstract communication scenario

While the black links between centre-side components and field devices may use a variety of communication protocols, mostly depending on the physical link conditions, the vast majority of other coloured links between centre-side components, internal to one organization or external to others, are

based on an internet protocol (IP) network and mostly use the transmission control protocol (TCP) transport layer protocol (user datagram protocol, UDP, is also possible in a few cases).

Nevertheless, as the different colours indicate, they can very well have significantly different requirements. Internal links (blue) can reside in one domain of trust, hence do not require protocols compatible with security gateways. This can already be different for links to other road operators (red) and will certainly not hold for links to other types of organizations, like service providers, via the Internet (green).

While different security requirements offer the most striking and obvious example, there are more criteria that can lead to different preferences on different types of links, e.g. scalability, robustness and integration complexity.

In broad terms, the colours blue – red – green form a hierarchy from more internal, closely-coupled, well-integrated systems towards external, loosely-coupled and non-integrated systems. The world of information and communication technology (ICT) offers a broad range of solutions for these different scenarios, offering different advantages and disadvantages. It is clear that the one-size-fits-all principle will not provide the most efficient way of working here. Even on the highest level of abstraction and inside the ICT domain itself, a well-known battle of paradigms between remote-procedure-call (RPC) type service specifications and RESTful architectures exists. The same clusters of options are found in the domain of ITS standards, where for example the European standard for the real-time information interface relating to public transport operations (SIRI; see the EN 15531 series) introduces both concepts as complementary options: Publish-Subscribe and Request-Response.

Furthermore, the ITS station architecture is not in contradiction with this document but is complementary to what is defined in this document. According to the principles and the taxonomy defined in ISO 21217, this document defines a conceptual notion of:

— How two central ITS (sub-)stations could communicate to:

  — deliver information (application data units);

  — negotiate functional service behaviour for collaborating traffic management functions (even if this use case could not directly be matched to ISO 21217 as it is not about information delivery).

— How a central ITS (sub-)station could communicate to deliver information (application data units) to another ITS station with the characteristics of a central ITS station.

This document specifies the process of defining the exchange characteristics by use case-driven feature selection of relevant parameters for the relevant OSI layers as defined in ISO 21217. Two exchange schemas are considered: information delivery and functional service negotiation between central ITS stations.

The drafting of this document was guided by the following principles:

— interoperability, such that different implementations can successfully engage in a data exchange process;

— supporting of legacy implementations which are based on existing (exchange) specification, in order to maximize investments already made by stakeholders;

— addressing other user profiles, not only road operators, thus making this document available to a broader audience;

— reusing existing (communications) standards, in order to reduce implementation complexity and take benefit of proven and already existing solutions for common ICT problems;

— maintaining a clear separation between the payload content and the exchange model.

Annex A details the adopted methodology for defining this exchange platform-independent model (PIM).

Intelligent transport systems — Data interfaces between centres for transport information and control systems — Platform-independent model specifications for data exchange protocols for transport information and control systems

1 Scope

This document defines and specifies component facets supporting the exchange and shared usage of data and information in the field of traffic and travel.

The component facets include the framework and context for exchanges, the data content, structure and relationships necessary and the communications specifications, in such a way that they are independent from any defined technical platform.

This document establishes specifications for data exchange between any two instances of the following actors:

— Traffic information centres (TICs);

— Traffic control centres/Traffic management centres (TCCs/TMCs);

— Service providers (SPs).

This document can also be applied for use by other actors, e.g. car park operators.

This document includes the following types of information:

— use cases and associated requirements, and features relative to different exchange situations;

— different functional exchange profiles;

— abstract elements for protocols;

— data model for exchange (informational structures, relationships, roles, attributes and associated data types required).

In order to set up a new technical exchange framework, it is necessary to associate one functional exchange profile with a technical platform providing an interoperability domain where plug-and-play interoperability at a technical level can be expected. The definition of such interoperability domains is out of scope of this document but can be found in other International Standards or Technical Specifications (e.g. the ISO 14827 series).

This document is restricted to data exchange. Definition of payload content models is out of the scope of this document.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
business scenario
high-level description of the interactions that can exist within a system being analyzed or between the system and external entities (called actors) in terms of business functions

Note 1 to entry: See also *use case* (3.21).

3.2
client
entity that receives the information

Note 1 to entry: It is represented in the information delivery *business scenario* (3.1).

3.3
exchange pattern
EP
basic exchange architecture template, described by UML communication diagrams, that identifies the actors in the exchange framework and the available interactions among them, which enable data exchange functionalities as a set of exchange features

Note 1 to entry: Exchange pattern interactions can be described by means of UML sequence diagrams and state machine diagrams in such a way that message-triggering conditions are fully identified and defined alongside any state update based on the subsequent interaction, i.e. exchanged messages and interaction-derived conditions.

3.4
collaborative ITS service
CIS
ITS service (3.7) that can be enabled by combing different "ITS services" that are provided by the combined effort of two to more stakeholders who can have different roles

EXAMPLE Traffic management centres, traffic information centres, service providers.

3.5
functional exchange profile
FEP
selection of data exchange features for a particular *business scenario* (3.1)

3.6
interoperability domain
pair of *functional exchange profile (FEP)* (3.5) and platform selected for implementing a data exchange subsystem

Note 1 to entry: Each *platform-specific model (PSM)* (3.11) document defines an interoperability domain, which ensures that two implementations of this PSM are interoperable and can successfully exchange *payload* (3.9).

3.7
ITS service
processing of information to address specific ITS requirements and implement ITS features such as to manage traffic or deliver information

3.8
payload content model
content model
UML definition of the data structures that can be used to describe travel and traffic information to be exchanged in an exchange system

3.9
payload publication
payload
bundle of information that is exchanged between two exchange systems containing an instance of the *content model* (3.8)

3.10
platform-independent model
PIM
document describing the abstract model of the standardized data exchange process in a platform-independent way

Note 1 to entry: This definition is specific to this document.

3.11
platform-specific model
PSM
document providing the implementation details of a *functional exchange profile (FEP)* (3.5) described in a *platform-independent model (PIM)* (3.10) for a concrete platform

Note 1 to entry: This definition is specific to this document.

3.12
profile-to-platform mapping
act of defining an *interoperability domain* (3.6)

3.13
pull exchange
exchange pattern (EP) (3.3) where the exchange of information is originated by the *client* (3.2)

3.14
push exchange
exchange pattern (EP) (3.3) where the exchange of information is originated by the *supplier* (3.20)

3.15
simple push
push-based *exchange pattern (EP)* (3.3) that does not require state to be maintained

3.16
snapshot
set of data providing all of the last known state as opposed to providing partial changes

Note 1 to entry: This definition is specific to this document.

3.17
snapshotpull
pull-based *exchange pattern (EP)* (3.3) where only the last *snapshot* (3.16) version is exchanged

3.18
snapshot push
push-based *exchange pattern (EP)* (3.3) where only the last *snapshot* (3.16) version is exchanged

3.19
stateful push
push-based *exchange pattern (EP)* (3.3) where data describing a communication session is maintained across successive communication within that session

3.20
supplier
entity that provides the information

Note 1 to entry: It is represented in the information delivery *business scenario* (3.1).

3.21
use case
UC
set of operational interactions between entities (called actors) and a system to ease understanding of the main functions behind such interactions

4 Symbols and abbreviated terms

ASN.1	Abstract Syntax Notation One
BUC	business use case
F&L	freight and logistic
HTTP	hypertext transfer protocol
ICT	information and communication technology
IP	internet protocol
ITS	intelligent transport systems
LOS	level of service
MDA	model-driven architecture
MMI	man-machine interface
pub/sub	publish-subscribe pattern
REST	representational state transfer
RPC	remote procedure call
SOAP	simple object access protocol
SSL	secure sockets layer
TCC	traffic control centre
TMC	traffic management centre
TIC	traffic information centre
TIS	traffic information service
TCP	transmission control protocol
TLS	transport layer security
TMP	traffic management plan
UDDI	universal description discovery and integration
UDP	user datagram protocol
UML	unified modeling language (see the ISO/IEC 19505 series)
VMS	variable message sign

W3C	world wide web consortium
WSDL	web service definition language
WSIL	web services inspection language
WSS	web services security
XML	extensible markup language

5 Exchange modeling framework

5.1 Overview

The model-driven approach is chosen to describe exchange: this leads to describing exchange systems by means of abstract models, named platform-independent models (PIMs), in which the modeling of exchange features is achieved by describing interactions among systems and subsystems as exchange patterns (EPs). These interactions implement system capabilities as features that fulfil exchange requirements requested by specific business scenarios which are used to describe specific uses of exchange.

5.2 Business scenarios and functional exchange profiles

This document is based on business scenarios, i.e. a high-level description of the interactions that can exist within a system being analyzed or between the system and external entities (called actors) in terms of business functions. Business scenarios are derived from application requirements on useful business information required and on technical capabilities enabled by available technologies. FEPs are identified to ensure interoperable services with the restriction of determining one FEP per business scenario for a specific EP, which is an abstract model of available technical platforms.

One business scenario can be supported by more than one FEP. FEPs can be enabled by several EPs (Figure 2).

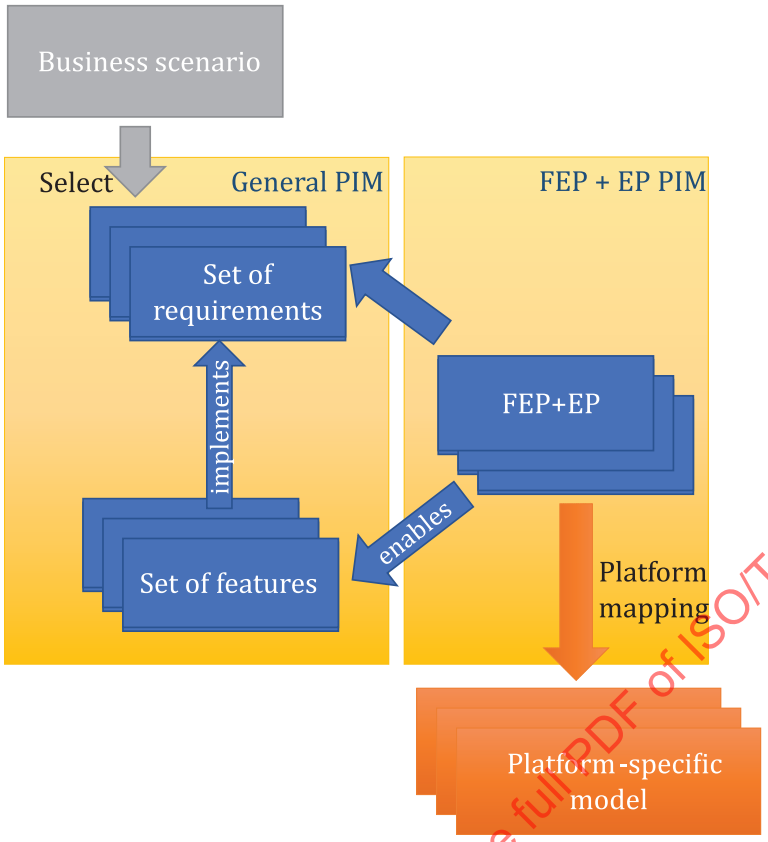

Figure 2 — Business scenario and functional exchange profiles

This document addresses the following business scenarios:

— information delivery,

— collaborative ITS services.

Other business scenarios can be developed in future editions of this document using the same methodology.

5.3 Requirements, features and exchange patterns

Requirements can vary depending on data exchange applications (i.e. use cases to be fulfilled). There are therefore many reasons to consider or ignore any requirement based both on the gathering of data at the supplier system and the usage of the delivered data by the client.

Requirements address the following items:

— information provision,

— communications,

— security,

— financial aspects.

Exchange is defined through enabling features which fulfil data exchange requirements.

Many possible technical EPs are possible, each of which can enable a subset of requirements in different ways. To be interoperable, a client and a supplier shall implement the same platform with the same pattern. Allowing a wide variety of possible EPs, the best suitable option for a use case application is chosen in order to fulfil the necessary set of requirements.

Based on the requirements of a specific business scenario selected for application use case implementation, a set of appropriate exchange features shall be combined into an FEP.

The following schema in Figure 3 represents the domains of PIM and PSM introducing EPs and FEPs.

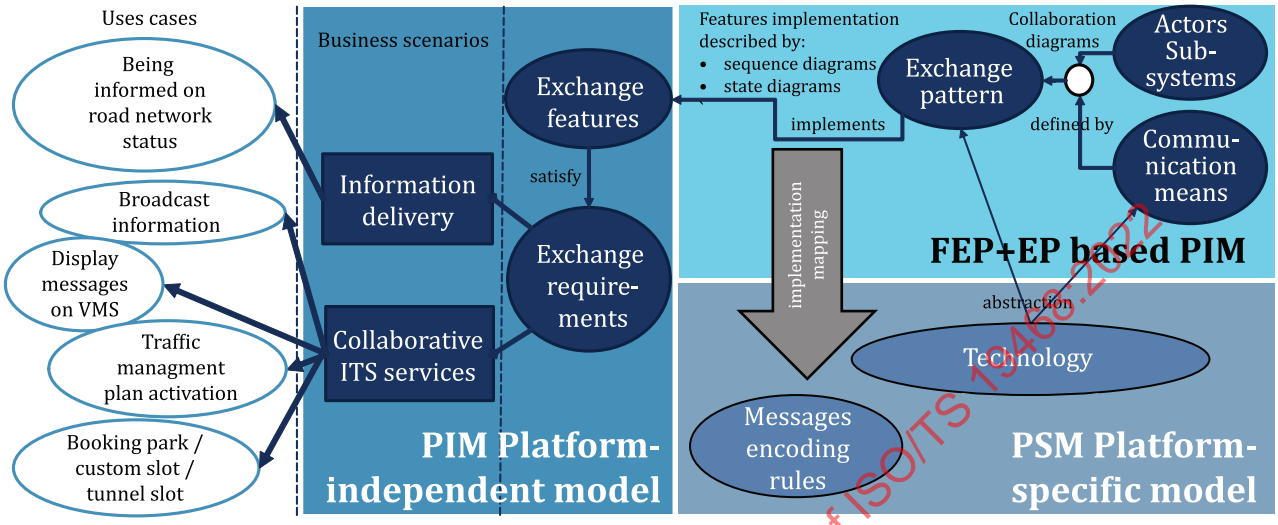

Figure 3 — Business scenario and functional exchange profile (FEP)

5.4 Business scenario: information delivery

5.4.1 Overview

One of the most common applications of a data exchange system is the exchange of traffic and travel information between two nodes. In such a scenario, one node acts as the supplier of the information while the other acts as the intended receiver of that information, i.e. the client.

EXAMPLE This is achieved by using the form of publications, e.g. in the European DATEX II.

The data delivery business scenario considers the actors as defined in Figure 4:

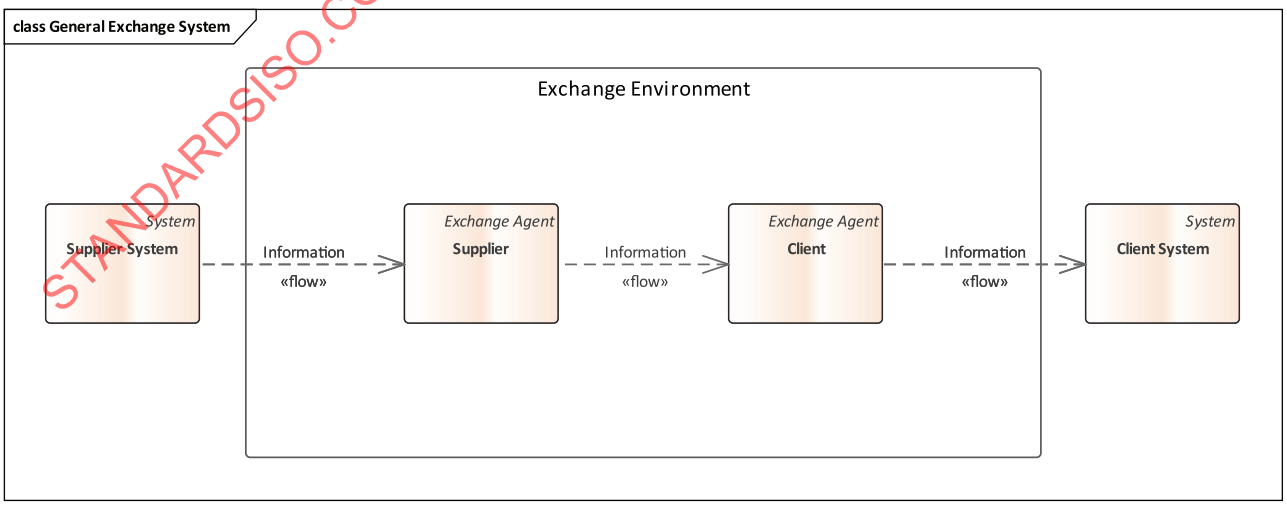

Figure 4 — General data delivery business scenario actors

Table 1 provides the basic definitions for exchange:

Table 1 — Main definitions in exchange

Name	Definition
Supplier system	A system which gathers information (road information) which needs to be conveyed to another system, named "client system". Examples of supplier systems are traffic control centres, traffic information centres or service provider systems, gathering road data from any available source they have.
Client system	A system which needs to update its internal information based on information which is available from another system, named "supplier". Examples of client systems are traffic control centres, traffic information centres or service provider systems.
Exchange environment	The set of components which enables information exchange among client systems and supplier systems via a data exchange protocol.
Supplier	The component of the exchange environment which is devoted to providing data to the client and retrieving them from supplier system.
Client	The component of the exchange environment which is devoted to collecting data from the supplier and delivering them to the client system.

Road and traffic information is gathered in a system named “supplier system”. In case this information is needed for any purpose by another system, named “client system”, it has to be transferred among the two systems by the exchange environment.

The data delivery business scenario describes the EP and messages which are needed to be exchanged among the supplier and client systems besides the underneath technology and EP. The purpose for which information is exchanged is not considered in this use case description.

As explained in the information delivery background in Annex F, any update of information status at the supplier system shall be replicated to the client system via information delivery. The main objective of information delivery is that information on the client system is updated exactly in the same way as it is in the supplier system without any difference in information values and semantics.

“Exchange message” is defined as the data structure in which the information is coded to transfer information in the exchange system from the supplier to the client.

Assuming S_c = Client status and S_s = Supplier status, exchange is a means to achieving an S_c equivalent to S_s.

Formally:

$S_s \rightarrow$ Information $\rightarrow$ Supplier $\rightarrow$ Exchange message $\rightarrow$ Client $\rightarrow$ Information $\rightarrow S_c$

This is to say that client system status is updated in an equivalent mode to supplier system status by means of data delivery exchange messages between supplier and client.

The information delivery business scenario scope is implemented by selected exchange features. These selected exchange features enable this scope and other secondary requirements which are based on available features on considered platforms and patterns.

1) Supplier system characterization:

   — shall provide data as input to the data exchange environment;

   — is mandatory for the information data delivery process.

2) Data exchange environment:

   — shall be an environment supporting the exchange of information and data by mean of messages;

   — the supplier of the data exchange system shall produce and transmit the messages (notification);

   — the client of data exchange system should receive and process the messages.

3) Client system characterization:

— shall receive traffic and travel-related data from the data exchange environment;

— can be either another system for further processing or a simple client for content visualization (the purpose of information exchange is not considered to be relevant in this information delivery business scenario);

— is mandatory for the information data delivery process.

5.4.2 Requirements

Annex B provides a description of all requirements that apply to specific business scenarios, including information delivery, whether actually used in a particular FEP or not. All requirements are organized into groups based on their characteristics.

NOTE A FEP is a different concept from content profile. The description of the content profiles that can exist in an information delivery scenario is not within the scope of this document. Content profiles are handled depending on the information exchanged.

5.4.3 Data delivery exchange pattern

For data delivery, the concept of EP is introduced, as well as the concepts PIM and FEP (see Clause 3 for definitions).

Examples of EPs are the GET method of HTTP, Pull, Push, PubSub, etc.

Once an EP and a selection of features (FEP) which can be implemented on this EP are set, a set of specification at PIM level for EP+FEP is well-defined.

A PIM for EP+FEP that enables all mandatory requirements is a valid platform for the corresponding business case (information delivery or collaborative ITS services).

5.4.4 Specific exchange pattern specification PIMs included in this document

In the following clauses, PIMs for EP/FEP will be described for the following platforms:

— Snapshot Pull;

— Snapshot Push;

— Simple Push;

— Stateful Push;

Annex E details the features of the main FEPs considered in this document for the corresponding EP.

5.5 Business scenario: collaborative ITS services

5.5.1 Overview

A service refers to any processing of data which enables a value for the data themselves. Within the field of ITS, ITS services can be considered as the processing of information to address specific requirements, such as to managing traffic and delivering information.

Collaborative ITS services (CISs) are ITS services that can be enabled by combining various ITS services, provided by several stakeholders who may have different roles (e.g. traffic management centres, traffic information centres, service providers).

CIS exchange specifications explore user requirements and define common techniques to address them in order to implement collaborative “ITS services” by different centres. They are based on exchanging

information to be processed by the different nodes and receiving the processing outcomes (feedback), giving a simple mechanism, upon which it could be possible to build more sophisticated workflows, enabling the coordination of operations distributed among many centres.

In the CIS business scenario, the exchange of information among centres is considered as a base mechanism for triggering specific processing and for providing services on this data, so the data exchange layer is to be considered as an ITS services enabler.

A description of ITS services is out of the scope of this document: a detailed description of possible usage of CIS is provided in Annex G.

Examples of ITS services in the different ITS domains are:

— TIS (traffic information services):
  — information delivery,
  — allowed channels.
— TMS (traffic management systems):
  — hard shoulder running,
  — dynamic speed,
  — dynamic lane management,
  — etc.
— F&L (freight & logistic):
  — secure truck parking,
  — etc.

5.5.2 Data exchange-enabling service request and feedback paradigm

The involved systems, which in data delivery had been considered as supplier and client, can be considered in this paradigm respectively as service requester and service provider.

At application level, for implementing a business case, such as management of TMP, workflow management is usually requested; this could be a simple workflow (one shot request and acceptance or rejection) or a more complex one: complex workflow modeling at application level is out of the scope of this document. This document provides the essential tools as building blocks for implementing such a simple workflow in order to raise a service request and provide feedback as exchange of payload to be processed and as processing results.

CIS-enabling mechanisms include:

— service request delivery from service requester to the involved service providers;

— service feedback delivery from service provider to the relative service requester.

The main characteristics of CISs are as follows:

— collaborative: two to many systems interact to achieve a common objective by processing data and to exchange processing results;

— CIS modeling considers the use of a set of "1 to *n*" exchange connections. Any single CIS triggering request is considered as one service requester and one to many service providers, so that any complex interaction can be considered a combination of single usages from multiple requesters;

— possible lock of resources at service provider sides (to be managed at application or human operator level) are not described in this document.

5.5.3 Requirements

Annex B provides a description of all requirements that apply to specific business scenarios, including collaborative informative services.

5.6 Exchange data model

To implement some exchange features (such as session management or link monitoring) or security features, extra information is to be added to the informational (payload) content. This extra information, named exchange information, enables messages to convey information and data which are related to client and supplier status, identity, authorization, etc.

Exchange information can be different among EP+FEP selection, but some common general information models are considered. These can be specialized to manage specific EPs and PIMs.

A general UML model for managing a minimum set of information for exchange is provided in C.2.

5.7 Data exchange features

5.7.1 Context diagram

This document defines the features and rules that systems shall implement in order to be able to communicate in the traffic and travel data exchange world.

The context diagram in Figure 5 shows the entities and the features specified in this document and which need to be addressed by the technical implementations. The diagram presents the features in different layers for application, message rules and communication:

— The “Application layer” is used for defining, using and implementing different business and application needs. This document describes the features to deal with how and when the information is published and made available, how subscriptions are managed and how to handle services and transactions. This layer defines the semantics of the communication.

— The “Messages rules layer” defines the features and the rules used for the transport of the messages. This is the layer where the rules (protocol) are defined that enable different systems (suppliers and clients) to communicate and understand each other, i.e. for sending and receiving messages. This layer defines the syntax of the communication.

— The “Communication layer” deals with the support for communication between systems, and defines the protocols and services used by the data exchange applications, e.g. TCP/IP, security, web services. The communication layer deals with rules at the lowest level, i.e. the physical exchange. This layer provides solutions to the defined requirements and features, although it is up to the upper applications to define which protocols to use and how to use them. This layer defines the physical support for the communication.

Figure 5 shows the topics broken up into boxes representing the exchange features and the relation with each layer.

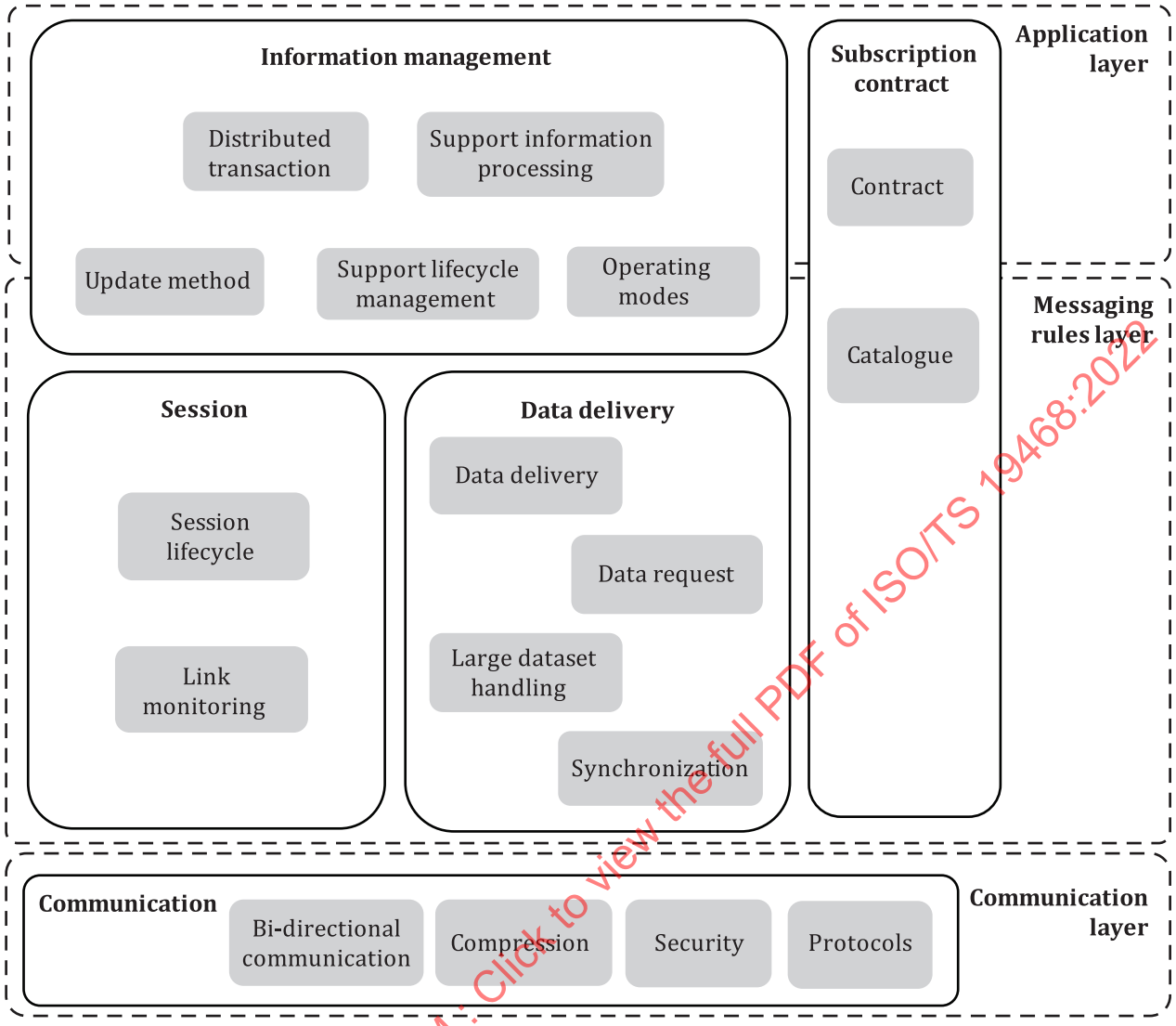

Figure 5 — Context diagram

5.7.2 Features

The features that support the requirements defined by the use cases specified in this document are detailed as follows:

1) The "subscription contract" feature group, detailed in Table 2, supports all features related to the contract or agreement, such as:

   a) Contract and contract life cycle: a model and features that can be used for the support of the information of a subscription contract;

   b) Catalogue: a model for handling catalogues.

The subscription contract is optional and includes the following features:

Table 2 — Subscription contract: Features and requirements

Features	Requirement type	Requirement name
Contract	Information	Subscription
		Client profiles
		Filter handling
Catalogue	Information	Catalogue exchange

2) The "session" feature group, detailed in Table 3, supports all features related to the establishment of a logical session.

   a) Session life cycle: features for managing the life cycle of a logical session (create, manage and terminate);

   b) Link monitoring: features for link monitoring and control.

The session feature group is optional and includes the following features:

Table 3 — Session: Features and requirements

Features	Requirement type	Requirement name
Session life cycle	Communication	Error handling
		Time-out management
		Session
Link monitoring	Communication	Error handling
		Time-out management
		Full reliability
		Link monitoring and control

3) The "information management" feature group, detailed in Table 4, handles the features related to the management of the information and includes features such as:

   a) "Operating modes": features to specify what portion of the information shall be exchanged;

   b) "Update methods": features that let a data exchange system specify when the information should be exchanged;

   c) "Life cycle management": features for handling the life cycle management of exchanged payload information for payload for which life cycle is applicable:

      i) Situation life cycle management

      i) Filter handling

   d) "Support information processing": features for handling directives to process exchanged data and send feedback on processing outcomes;

   e) "Distributed transaction": features for handling a transaction on several systems consistently, i.e. an operation is capable of maintaining data consistency among several systems based on undertaken operator actions.

The "information management" feature group includes the features listed in Table 4.

Table 4 — Information management: Features and requirements

Features	Requirement type	Requirement name
Operating modes	Information	Reference datasets for different versions
Update methods	Information	Full audit trail data delivery (all state changes)
Life cycle management	Information	Support for life cycle management
Support information processing	Information	Support for information management
		Support for feedback on information management
Distributed transaction	Information	Distributed transaction
		Distributed atomic transaction

4) The "data delivery" feature group, detailed in Table 5, supports all features to exchange data between the supplier and client. In the publish-subscribe pattern, this feature group will support all related interfaces between the producer and the consumer. It supports features such as:

   a) "Data delivery": features to delivery information by the supplier to the client on a push mode (direct delivery and fetched delivery);

   b) "Data request": features to exchange information requested by the client;

   c) "Large datasets": support the exchange messages with large volumes;

   d) "Synchronization": how to ensure data synchronization between the systems that are communicating.

The "data delivery" feature group includes features and requirements listed in Table 5.

Table 5 — Data delivery: Features and requirements

Features	Requirement type	Requirement name
Data delivery	Information	Extensibility
	Communication	Delivery/response
		Message sequence
		Snapshot data delivery (last known state)
		Exchange quality measures (e.g. response times-tamp)
		On occurrence update
		Periodic update
	Security	Client identification
		Supplier identification
	Information	Incremental data delivery

Table 5 *(continued)*

Features	Requirement type	Requirement name
Data request	Information	Extensibility
	Communication	Request/response
		Message sequence
		Snapshot data delivery (last known state)
		Exchange quality measures (e.g. response times-tamp)
		On occurrence update
		Periodic update
	Security	Client identification
		Supplier identification
Large datasets handling	Information	Data delivered as soon as possible
		Delayed delivery
		Multi-part data delivery
Synchronization	Information	Synchronization
	Communication	With state supplier
		Failed data recovery

5) The "communication" feature group, detailed in Table 6, handles all features related to a protocol (close to the physical layer). It is used by the application for the massage transport:

   a) "Communication": describe the communication protocols that can be used;

   b) "Security": describe how security features can be implemented;

   c) "Compression": how data compression can be used while transmitting data;

   d) "Bi-directional communication": enable a client to send back data to a supplier as feedback on data exchanged and processing status of data based on the supplier's processing directive in collaborative ITS services.

The "communication" function is responsible for implementing the following features:

Table 6 — Communication: Features and requirements

Features	Requirement type	Requirement name
Security	Security	Security (data)
		Integrity
		Confidentiality
		State the intended recipient
		Client authentication
		Supplier authentication
		Client authorization
		Non-repudiation
Compression	Communication	Compression
Communication	Communication	Timely responses
Bi-directional communication	Communication	Bi-directional communication

5.8 Exchange pattern modeling using UML

EPs which enable features for selected FEPs can be described by use of certain UML diagrams that describe which actors are involved in the exchange, and the specific interactions among them that enable the features themselves.

The exhange system description assumes that external systems like TCC and TMC are external to the exchange and that exchange agents are the actors who may implement the exchange role of client, supplier, service requester or service provider as defined in the business case.

The Figure 6 describes the main exchange-involved actors as well as other interacting actors external to exchange.

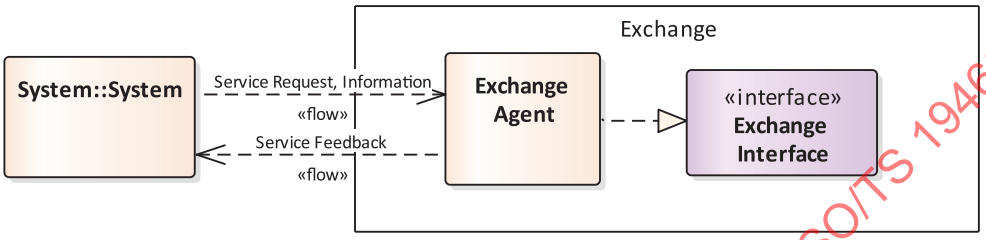

Figure 6 — Exchange actors

Systems (e.g. TCC or TMC system) may act both as client or supplier and for exchange purposes. Systems need to interact with the exchange agent of the same client or supplier type that realizes a system interface enabling the exchange features.

The relative system and interfaces can be specialized based on multiple options for a system to be a client or supplier for data delivery, or a service requester or service provider for collaborative ITS services business scenarios. Figure 7 and Figure 8 show the data delivery exchange actors and the data delivery exchange systems, agents and interfaces, respectively.

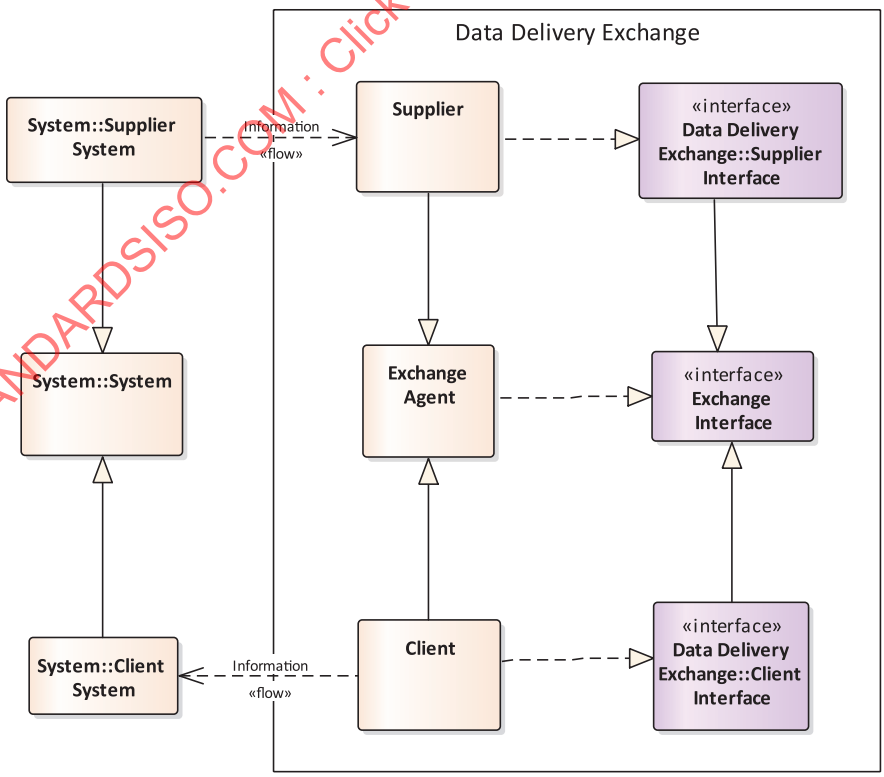

Figure 7 — Data delivery exchange actors definitions

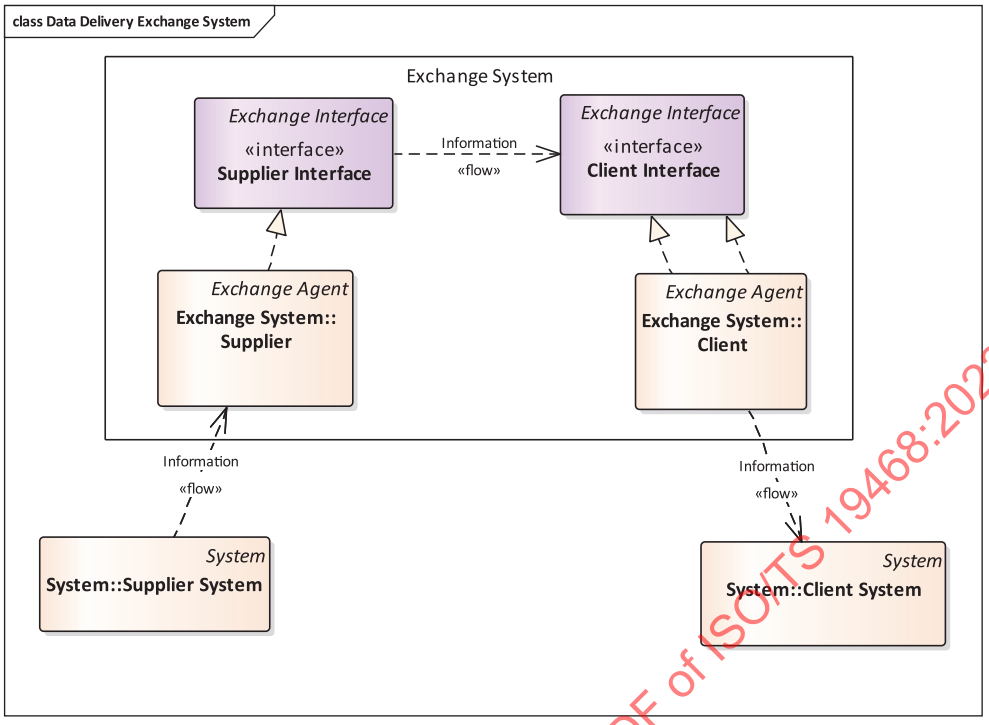

Figure 8 — Data delivery exchange systems, agents and interfaces

Figure 9 and Figure 10 show the systems and exchange agents and interfaces for collaborative ITS services.

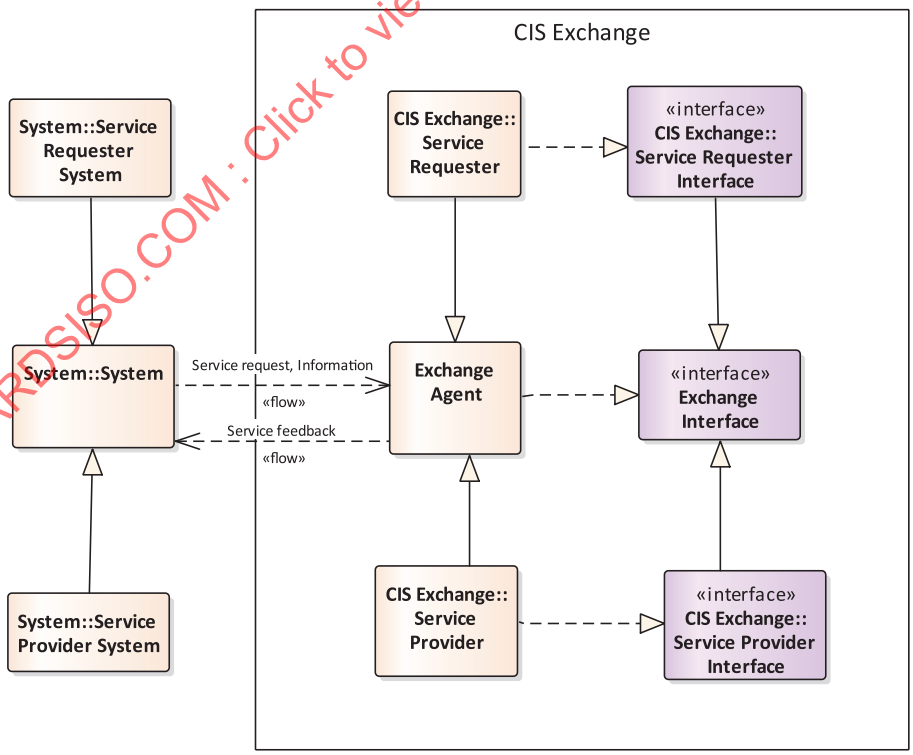

Figure 9 — CIS exchange agents definition

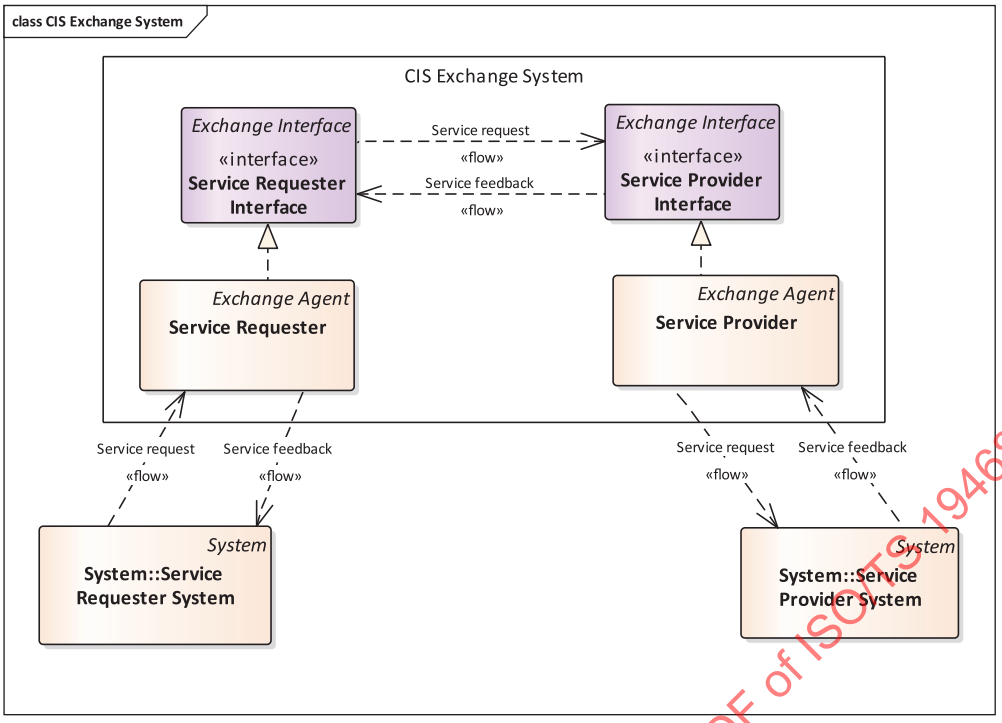

Figure 10 — CIS exchange systems, agents and interfaces

This system, exchange agents and interfaces classes will be used in the FEP+EP specification descriptions in the following clauses to describe the provided interfaces and interactions. Each exchange interface realization will be implemented as a specific specialization for its FEP+EP. For example, snapshot push will have its specific "snapshot push" supplier interface, i.e. the "snapshot push client interface", which will be described as a specialization of the exchange snapshot push client interface. The modeling for a snapshot push client system and its exchange agent and interface is shown in Figure 11.

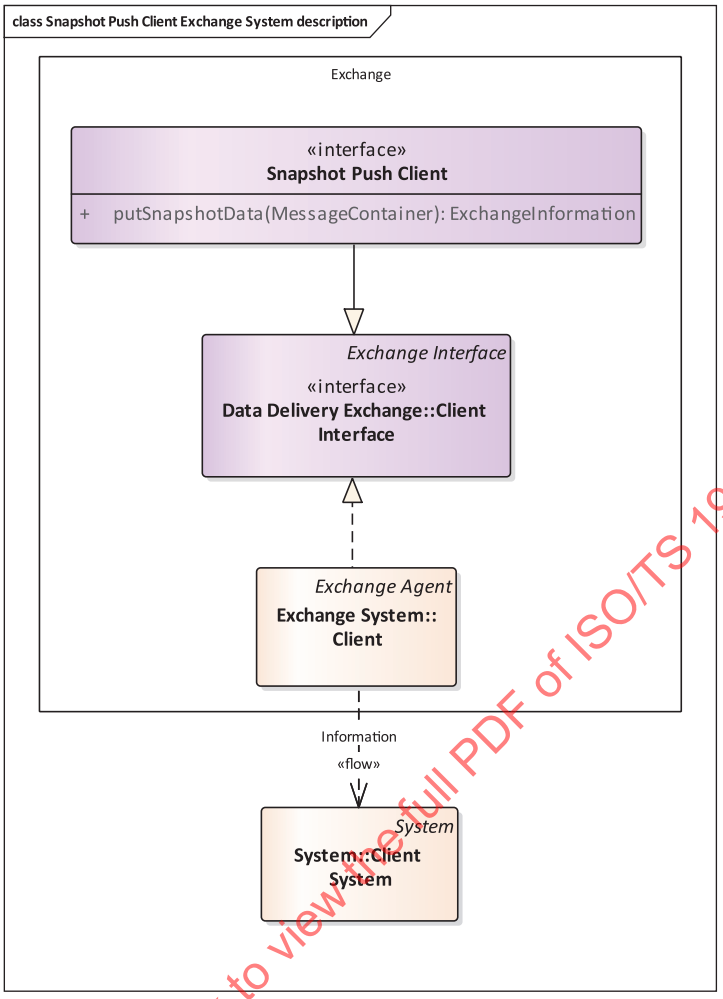

Figure 11 — Sample of "snapshot push" client system and its exchange agent and interface

The relevant communication diagrams will be introduced in the FEP+EP descriptions in the following clauses.

In the description of the interactions among exchange systems and subsystems, UML sequence diagrams will be used. Interactions among actors and their interfaces are described using the actor themselves. The realized specialized interfaces are not mentioned for schema understanding convenience. Figure 12 is an example of the provided sequence diagram.

NOTE In all sequence diagrams introduced in this document, only the interactions among supplier and client are mandatory, and they are intended for the interaction among supplier and client exchange interfaces. Other interactions are introduced for better understanding of the overall system workflows and can be assumed to be generally valid, but those interactions are defined and implemented in the ITS systems in a way which is not defined in this document.

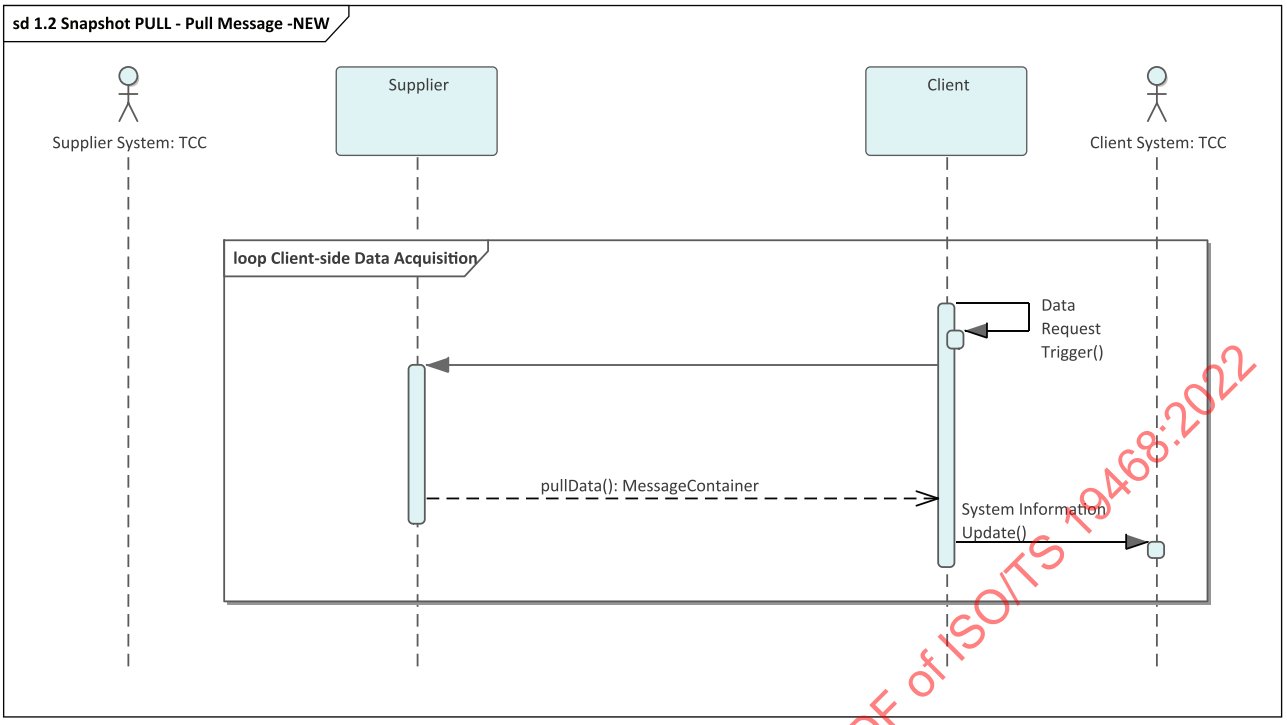

Figure 12 — Example of a sequence diagram

State diagrams can also be used to specify the specific status of exchange operations which will lead to specific interface behaviours such as return messages definition and specific interactions provided (e.g. synchronization, synchronization request). State diagrams are not needed for understanding in simple cases such as stateless FEP+EP so they will not be described in some FEP+EP descriptions.

6 Snapshot pull

6.1 Overview

The “snapshot pull” EP+FEP at PIM level is based on information retrieval by a client from a supplier which delivers a snapshot of information (i.e. all currently available information content) to the client. It can be implemented in several platforms: some examples are XML retrieval of generated XML files by http/get, or supplier exposing a SOAP Web Service method (e.g. named "PULL"), from which currently available data is retrieved by the client.

This “snapshot pull” does not manage session life cycle and link monitoring requirements, as well as synchronization. This feature and related requirements are not considered in this pattern. Synchronization is not required as implicit when delivering snapshots of currently available information content.

To describe the snapshot pull EP+FEP at PIM level, all features are described in a general abstract format, independently from the specific technology platform with which this model will be implemented (e.g. http/get XML, WebService). Table 7 shows a selection of features for snapshot pull.

Table 7 — Selection of features for snapshot pull

Features area	Feature	Snapshot pull implemented
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	N

Table 7 *(continued)*

Features area	Feature	Snapshot pull implemented
	Link monitoring	N
Information management	Operating modes	Periodic or On Occurrence (i.e. triggered by client conditions)
	Update methods	Snapshot
	Life cycle management	Y, based on snapshot: new and updated content delivered, outdated data not delivered
Data delivery	Data delivery	Y
	Data request	N
	Large datasets handling	N
	Synchronization	Y
Self-Description	Handshake	N
Communication	Security	To be defined at PSM level
	Compression	To be defined at PSM level
	Communication	To be defined at PSM level

6.2 Exchange pattern messages definition

6.2.1 Overall presentation

The information delivery business scenario description and definition states that data exchange is needed to align the information kept by the supplier system into the client system. For this purpose, an exchange system is used which provides tools enabling messages generation and their transfer between supplier and client (see Figure 13).

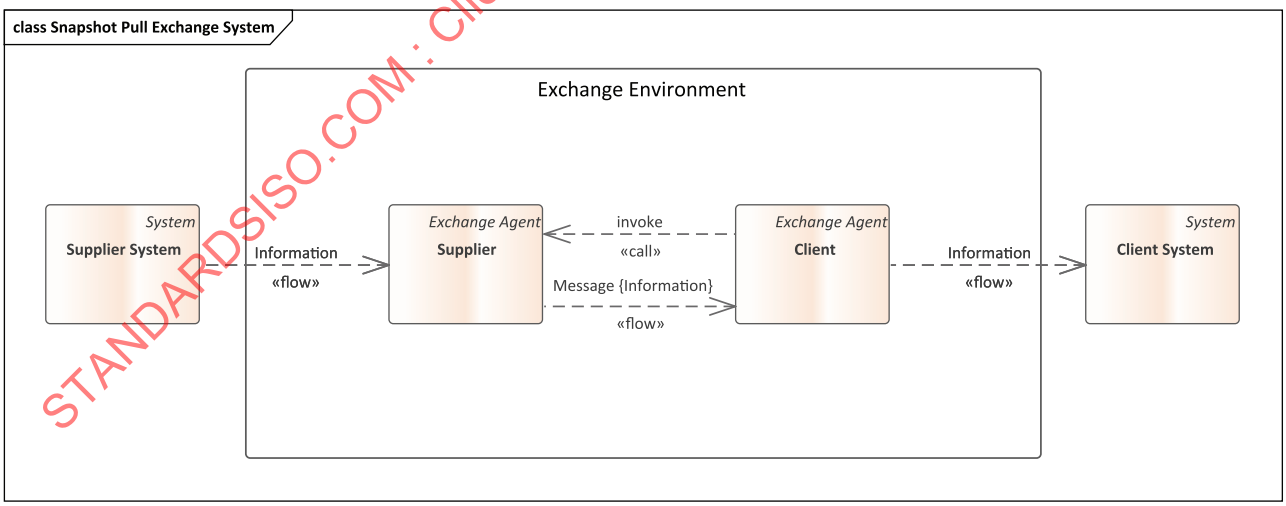

Figure 13 — Snapshot pull exchange actors

The snapshot pull EP is described in the following subclauses.

6.2.2 Exchange pattern definition

In a snapshot pull context, the supplier exchange system provides a mechanism to retrieve currently available and valid data (i.e. a snapshot of information) from an action taken at the client side, which will invoke this specific mechanism offered by the supplier.

In the context of the snapshot pull FEP +EP framework, to enable interoperability among client and supplier, all rules defined in this subclause apply.

A snapshot pull supplier exchange system shall realize a snapshot pull supplier interface that provides a "pullSnapshotData" method for implementing the snapshot pull mechanism.

A snapshot pull client exchange system shall realize a snapshot pull client interface that invokes the "pullSnapshotData" method provided by the snapshot pull supplier interface to retrieve snapshot data.

Figure 14 shows the communication diagram for snapshot pull FEP+EP.

In this FEP+EP framework the client “pulls” messages from the supplier.

The client shall deliver no information to the supplier in the pull request.

The supplier shall retun the client pull request by delivering a "MessageContainer" information which includes ExchangeInformation.

NOTE This return message is available to bring some information from the client to the supplier, (e.g. exchange information), which can be used for any further exchange features implementations or application level checks or processing which are out of scope of this FEP+EP specification.

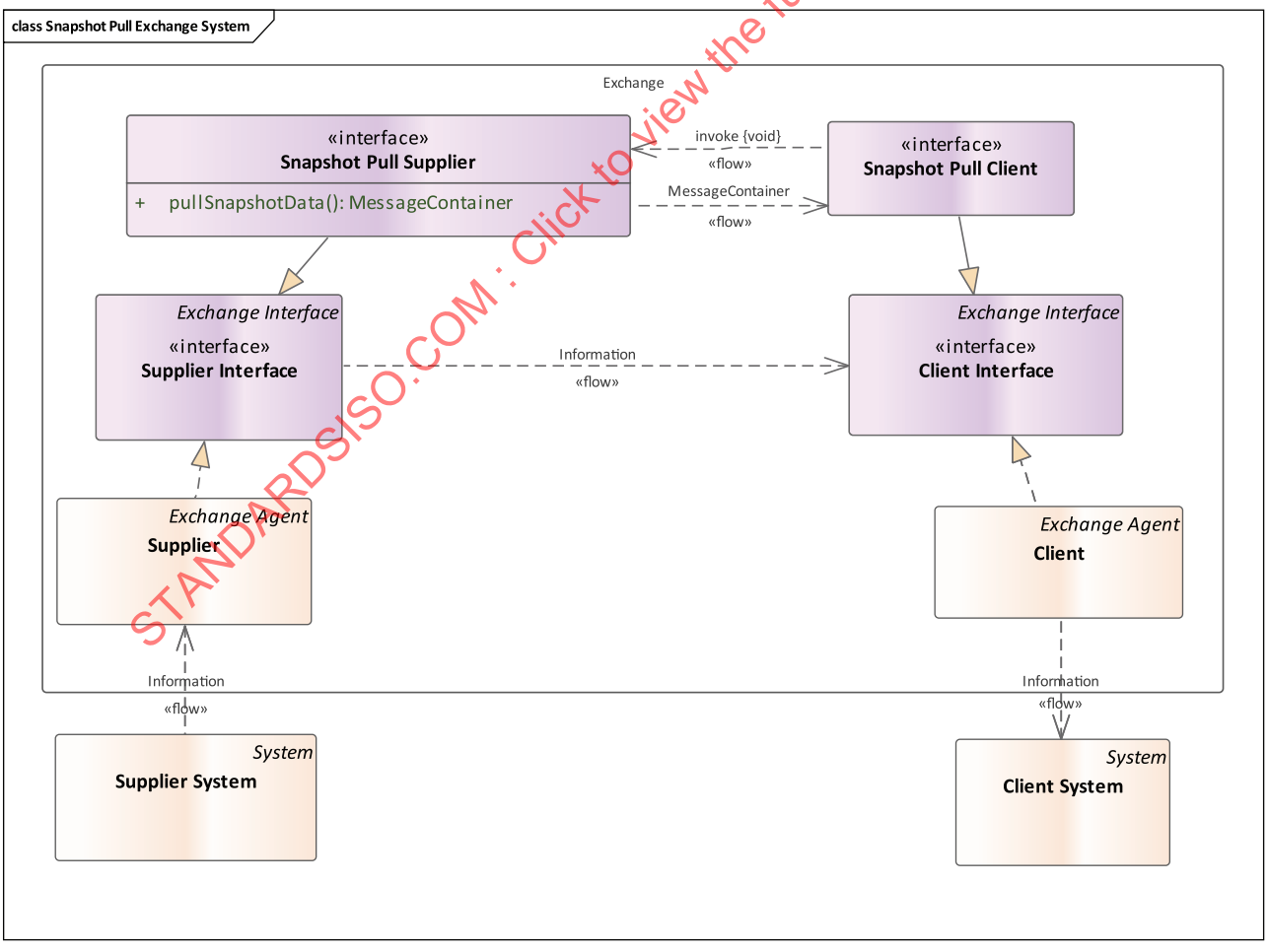

Figure 14 — Snapshot pull exchange subsystems, interface interactions and methods

The client takes the initiative to retrieve the data based on application level requirements which determine the needed exchange operating mode (e.g. on occurrence, condition-triggered or periodic).

6.2.3 Relevant exchange information in exchange data model

No exchange information is needed in this pattern to implement data delivery features. Nevertheless, a basic exchange data model (see in Annex C) has been provided to allow the implementation to deliver more than one payload content on the same message and further information to allow managing extra features not required by the snapshot pull exchange.

A “MessageContainer” instance should be retrieved using the basic exchange data model as reported in Figure 14. Alternatively, a payload may be retrieved without any further information.

Information related to exchange that should be managed to make application development easier is fully described in the basic exchange data model.

Related exchange context information is:

— Supplier-related information

  — Requirement: supplier identification.

Related dynamic information is:

— Exchange DynamicInformation (provided both by the client and the supplier) wraps information such as exchangeStatus ("online", "offline", "undefined", etc.)

— Message generation timestamp information

  — Requirement: reliable information.

6.2.4 Exchange messages

— Payload message:

  — Payload messages should be delivered wrapped into a container (the basic exchange data model in Annex C with exchange data applies).

  — Payload messages may contain a payload update timestamp which can be used to understand when the payload has been created/updated for error management and processing saving.

6.3 State diagrams

State diagrams are not needed and not developed for stateless FEP+EP as snapshot pull.

6.4 Features implementation description

6.4.1 Overview

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the snapshot pull exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

6.4.2 Subscription contract

6.4.2.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in the client to assess the identity of the supplier and authenticate the supplier request in message exchange.

6.4.2.2 Catalogue

Managed offline, not automated.

6.4.3 Session

6.4.3.1 Session life cycle

No session is managed for the current EP+FEP.

6.4.3.2 Link monitoring

Link monitoring is not managed for the current EP+FEP.

6.4.4 Information management

6.4.4.1 Operating modes

The available operating mode for client pull is Periodic or On Occurrence (i.e. condition-triggered based on client). Pull-exchange is based on client-side conditions.

6.4.4.2 Update methods

The available update method is snapshot, i.e. retrieval of only currently valid data.

6.4.4.3 Life cycle management

Currently available information is included in the payload at a supplier system to prepare message delivery. It can be done at time-out on a cycle basis or at a specific triggering condition as "data updated" condition.

For life cycle management information, a snapshot includes all active information. Outdated information is not delivered in content.

For sampled information, the snapshot information contains the last sampled data available at the supplier site.

Whenever a condition is raised, the supplier system triggers it to the supplier to manage the creation of a payload pull delivery message (see Figure 15).

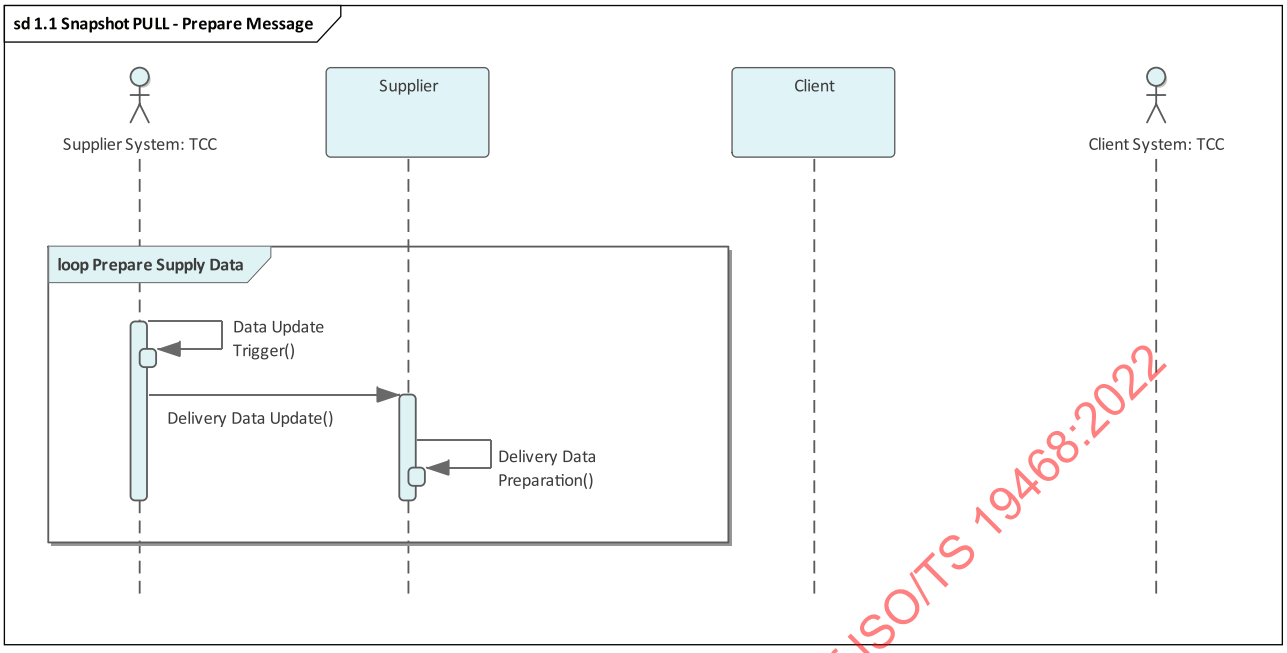

Figure 15 — Snapshot pull payload delivery creation: information management at supplier side

Information management for snapshot pull at client side is implemented as follows: when a client gets a snapshot of the last updated/created items it includes all valid active information. Information which had been delivered and which is not available in the last delivered payload shall not be considered after last devlivery, i.e. it has been invalidated either as closed or cancelled information.

6.4.5 Data delivery

6.4.5.1 Data delivery

The sequence diagram for data delivery is as follows in Figure 16:

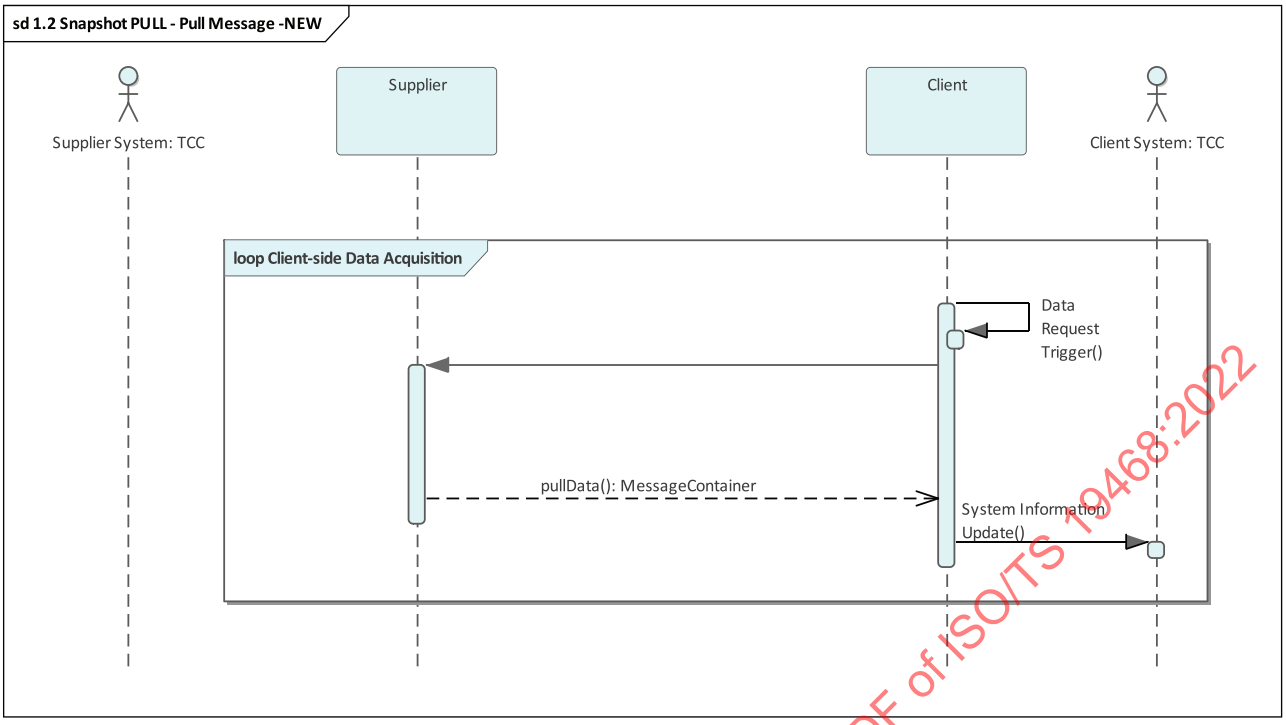

Figure 16 — Snapshot pull sequence diagram for data retrieval: implicit data delivery

When a pullSnapshotData request is triggered from the client to the interface method on supplier SnapshotPull interface, the corresponding snapshot payload(s) available shall be delivered as a return message by the supplier enclosed in a MessageContainer.

NOTE 1 Depending on implementation (e.g. http/get of XML static files generated at supplier side) the payload message is generated by the supplier based on conditions which are only managed by the supplier (e.g. event update or data gathered at the supplier side). In these cases, extra information can be available to implement some optimization, such as bandwidth saving, by not transferring the same data in case no update has been generated. This aspect will be described when applicable for PSM mapping.

NOTE 2 ExchangeInformation delivered in the return message by the supplier may contain any information which can be used to inform the client about client request processing which may be implemented in an optional pullRequest check. A processing for delivery based on client may be implemented by the supplier based on ExchangeInformation and offline subscription agreements. Any processing description for payload delivery based on client which can be implemented on supplier side based on any subscription agreement among client and supplier, are out of scope of this FEP+EP specification.

6.4.5.2 Data request

Not implemented in this pattern.

6.4.5.3 Large datasets handling

Not described in this pattern at PIM level (it may be implemented at PSM level as optimization; see 6.4.8).

6.4.5.4 Synchronization

Implicit synchronization is available as only currently available elements are retrieved by snapshot pull.

6.4.6 Self-description

Handshake is not available.

6.4.7 Communication

Communication features are implemented at PSM level. They are relevant for the specific platform chosen on which the EP will be implemented (e.g. http/XML, Web Services SOAP, REST, etc.).

6.4.8 General optimization issues

Some EP features of any context diagram features groups (e.g. information management, data delivery etc.) allow the implementation of general optimization such as processing saving and bandwith.

Payload timestamp information is available for client-side processing optimization made at the application level.

Pull message may be generated for all clients reducing processing resources at the supplier-side.

No extra optimization issues are considered in this EP+FEP.

7 Snapshot push

7.1 Overview

The "snapshot push" EP/FEP at PIM level is based on pushing information by the supplier to the client delivering a snapshot of information, i.e. all currently available information content, to the client. This EP can be implemented in several platforms: some examples are XML delivering of generated XML files by http/post, or a client exposing a SOAP web service method (e.g. named "PUSH") when currently available data can be sent by the supplier to the client.

This "snapshot push" does not manage session life cycle and link monitoring requirements, as well as synchronization. These features and related requirements are not considered in this pattern. Synchronization is not required as implicit when delivering snapshots of currently available information content.

To describe snapshot push EP/FEP at PIM level, all features are described in a general abstract format, independently from the specific technology platform in which this model will be implemented (e.g. http/get XML, WebService). Table 8 shows a selection of features for snapshot push.

Table 8 — Selection of features for snapshot push

Features area	Feature	Snapshot push implemented
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	N
	Link monitoring	N
Information management	Operating modes	Periodic or On Occurrence (i.e. triggered by supplier condition)
	Update methods	Snapshot
	Life cycle management	Y, based on snapshot: new and updated content delivered, outdated data not delivered

Table 8 *(continued)*

Features area	Feature	Snapshot push implemented
Data delivery	Data delivery	Y
	Data request	N
	Large datasets handling	N
	Synchronization	Y
Self-Description	Handshake	N
Communication	Security	To be defined at PSM level
	Compression	To be defined at PSM level
	Communication	To be defined at PSM level

7.2 Exchange pattern messages definition

7.2.1 Overall presentation

The information delivery business scenario description and definition imply that data exchange is needed to align the information kept by the supplier system into the client system; for this purpose, an exchange system is used which provides tools enabling message generation and their transfer between supplier and client (see Figure 17).

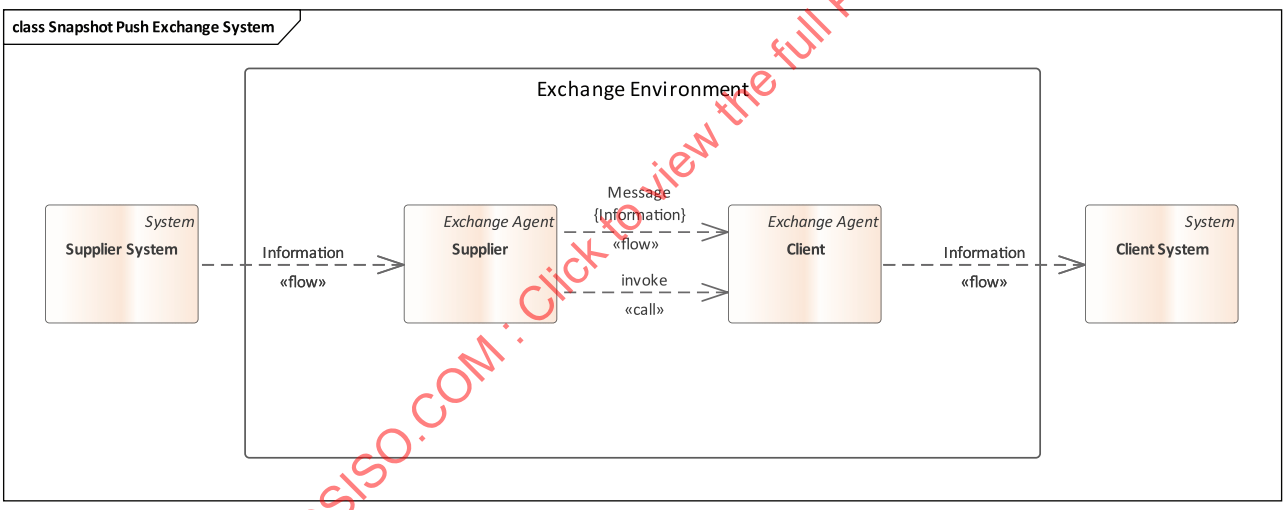

Figure 17 — Snapshot push exchange actors

The snapshot push EP is described in the following subclauses.

7.2.2 Basic exchange pattern

In a snapshot push context the client provides a mechanism to receive data from an action taken at a supplier site invoking specific resources / methods offered by the client.

The snapshot push client provides a mechanism to the snapshot push supplier to push currently available data, also called “snapshot” of information, i.e. current information at supplier system or last retrieved information for sampled data (see Annex F).

In the context of the "Snapshot Push" FEP +EP framework, to enable interoperability among client and supplier, all rules defined in this clause apply.

A snapshot push Client exchange system SHALL realise a snapshot push client interface wich provides a putSnapshotData method.

A snapshot push supplier exchange system SHALL realise a snapshot push Supplier interface which invokes the putSnapshotData Method provided by the snapshot push Client interface to deliver snapshot data.

Figure 18 shows the communication diagram for Snapshot Push FEP + EP.

In this FEP+EP framework the supplier "pushes" messages to the client.

The client shall acknowledge the received message by a return information to the supplier. This return information shall be coded as ExchangeInformation.

NOTE This return message is available to bring some exchange information from the client to the supplier which can be used for any further exchange features implementations or application level checks or processing which are out of scope of this document.

The supplier takes the initiative to deliver the data based on application level requirements which determine the needed exchange operating mode (e.g. on occurrence or periodic).

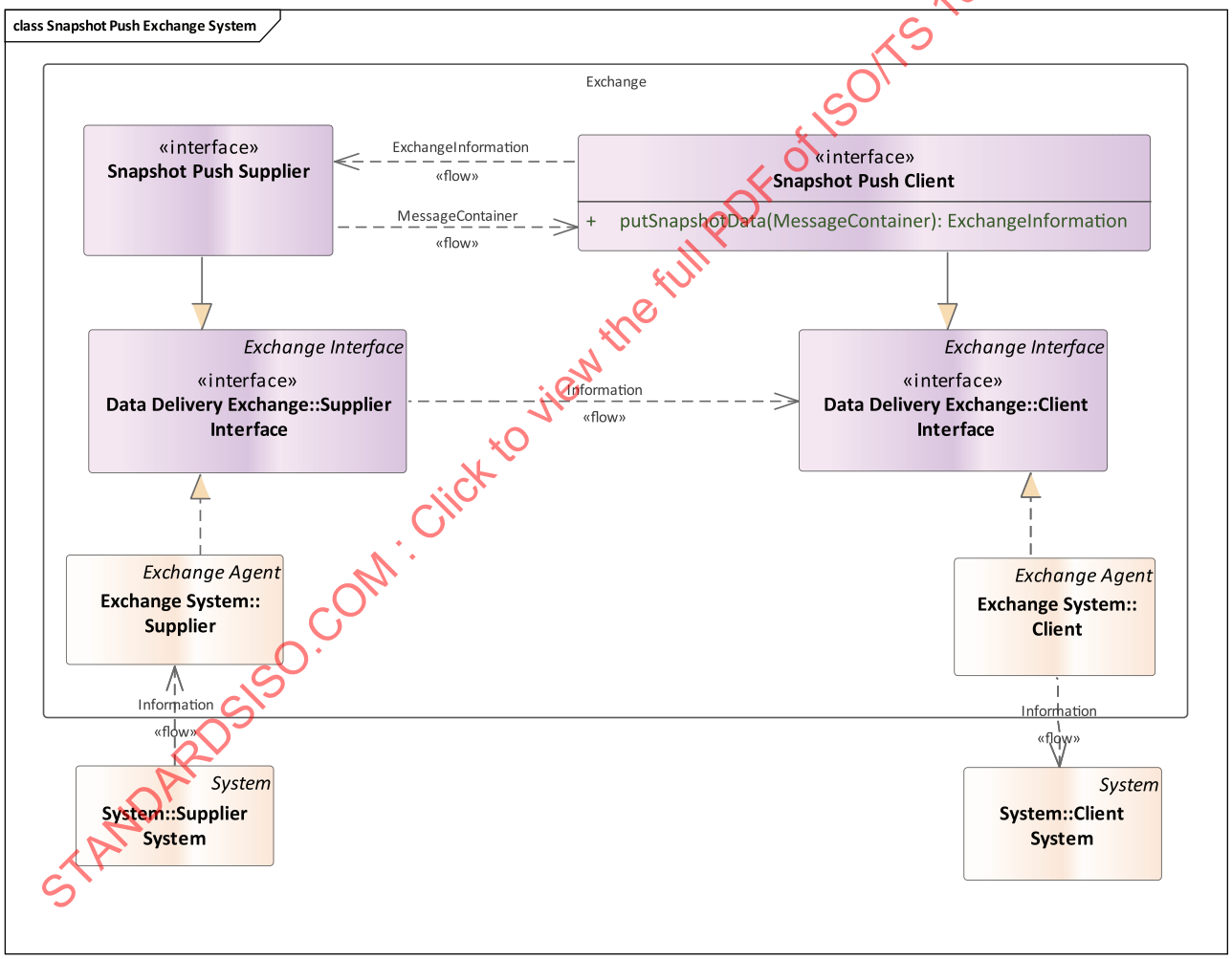

Figure 18 — Snapshot push exchange subsystems, interfaces interactions and methods

7.2.3 Relevant exchange information in exchange data model

No extra exchange information is needed in this pattern to implement any described features.

A basic exchange data model has been provided to allow the implementation to deliver more payload contents in the same message and further information to allow managing some extra features not

required by the basic snapshot push exchange. The usage of the exchange data model wrapping is for harmonization with other exchange patterns such as "simple push" or "stateful push".

A container should be retrieved using basic exchange data model as reported in the previous figure. Alternatively, a payload may be delivered.

An ExchangeInformation is returned to convey information about exchange operation and connection status.

Related exchange context information is:

— Supplier-related information

  — Requirement: supplier identification.

— Client-related information

  — Requirement: client identification.

Related dynamic information is:

— Exchange DynamicInformation (provided both by the client and the supplier) wraps information such as exchangeStatus ("Success", "Fail", "Close Session Request", "Snapshot Synchronization Request")

— Message generation timestamp information:

  — Requirement: timely, reliable information, session management.

7.2.4 Exchanged messages

— Payload message.

  — Simple payload messages can be exchanged within this FEP+EP.

  — Payload messages should be delivered wrapped into a container (see basic exchange data model in Annex C) with exchange data.

  — Payload messages contain payload update timestamp which can be used to understand when payload has been updated for error management and processing saving.

7.3 State diagrams

State diagrams are not needed and not developed for stateless FEP+EP as snapshot push.

7.4 Features implementation description

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the snapshot pull exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

7.4.1 Subscription contract

7.4.1.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in the client to assess the identity of supplier and authenticate the supplier request in messages exchange.

7.4.1.2 Catalogue

Managed offline, not automated.

7.4.2 Session

7.4.2.1 Session life cycle

No session is managed for the current EP+FEP.

7.4.2.2 Link monitoring

Link monitoring is not managed for the current EP+FEP.

7.4.3 Information management

7.4.3.1 Operating modes

Available operating mode for snapshot push is “Periodic”, or “On Occurrence” (i.e. condition-triggered based on supplier) push-based on supplier-side conditions.

7.4.3.2 Update methods

Available updated method is snapshot, i.e. retrieval of only currently valid data.

7.4.3.3 Life cycle management

Currently available information is included in the payload at a supplier system to prepare message delivery. It can be done at time-out on a cycle basis or at a specific triggering condition as “data updated” condition.

For life cycle management information, snapshots include all active information, outdated information is not delivered in content.

For sampled information, the snapshot information contains the last sampled data available at supplier site.

Information management for snapshot push is implemented as follows: when a client gets a snapshot of the last updated/created items, including all active items, it has to check for information which has been removed from the payload to deduce it has been invalidated (i.e. closed or cancelled).

7.4.4 Data delivery

7.4.4.1 Data delivery

For all clients with an active subscription, whenever a payload delivery condition is triggered, the supplier system shall manage the creation of a payload push message in a MessageContainer and shall deliver it by its SnapshotPush supplier interface to the corresponding SnapshotPush method available via the SnapshotPush client interface.

Figure 19 illustrates the sequence diagram which describes the interaction amongst the exchange supplier and its clients.

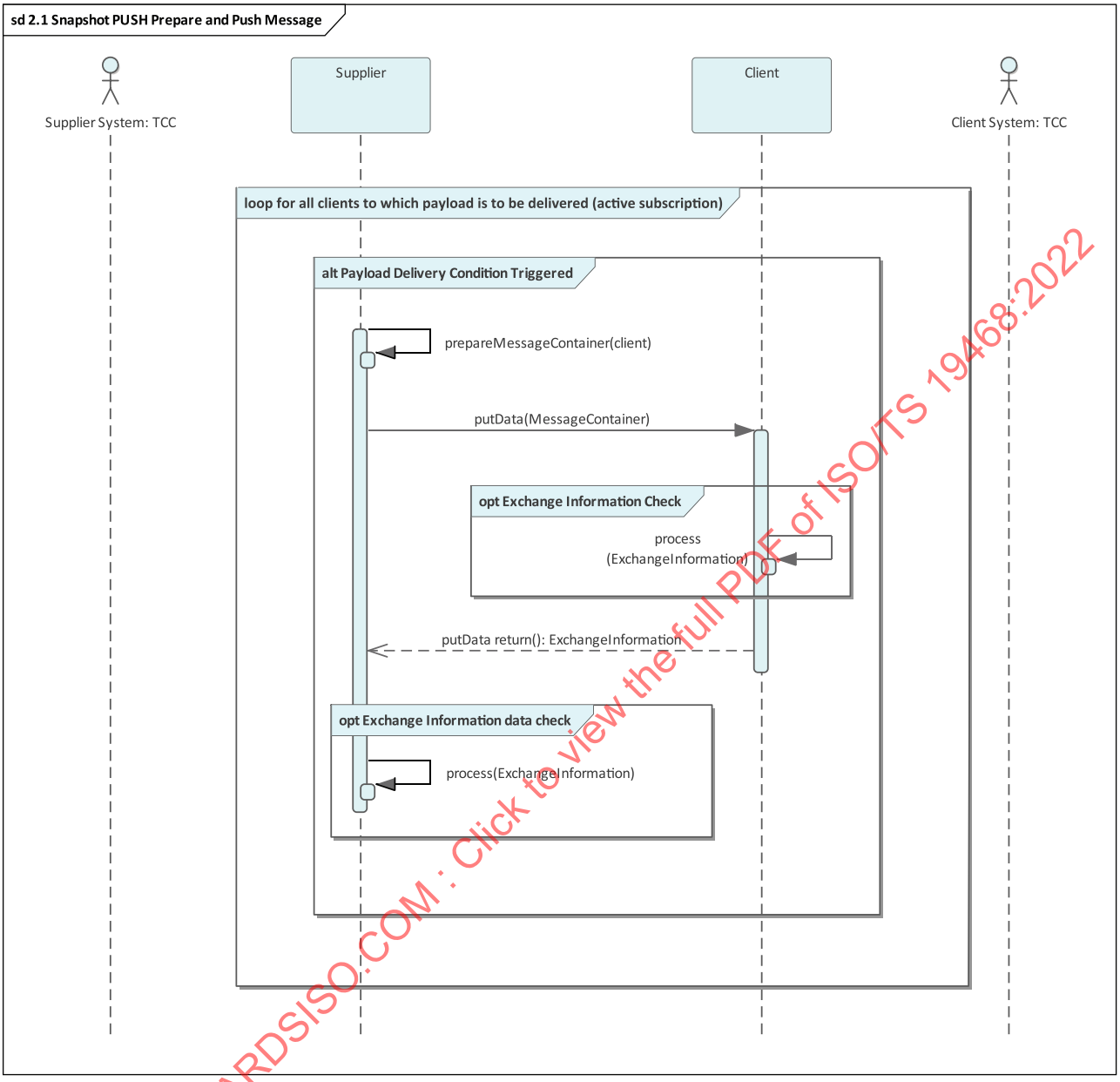

Figure 19 — Snapshot push sequence diagram: information management at supplier side

NOTE 1 Snapshot push message generation can create different a payload based on the client to which messages are to be delivered based on subscription information. Optional payload delivery processing based on ExchangeInformation and client subscriptions information are out of scope of this FEP+EP specification and are not described.

The client may verify the message delivered by the supplier and shall return an ExchangeInformation message to the supplier with information back to the supplier about the delivered message (i.e. return status and exchange status defined in basic exchange data model) which may be processed by the supplier to implement optional exchange checks.

NOTE 2 ExchangeInformation provided by the supplier and the client can optionally be checked to implement optional features enabled by the supplier and defined by the supplier and the client in the optional subscription process which is out of scope of this FEP+EP specification.

NOTE 3 ExchangeInformation delivered in the return message by the snapshot push client can contain any information which can be used to inform the snapshot push supplier about any client request processing, which can be implemented in an optional check. Optional processing to deliver payload information based on agreements among client and supplier can be implemented by the supplier based on offline subscription agreements. These processing descriptions and specifications are out of scope of this FEP+EP specification.

7.4.4.2 Data request

Not implemented in this pattern.

7.4.4.3 Large datasets handling

Not described in this pattern at PIM level. May be implemented at PSM level (see 7.4.7).

7.4.4.4 Synchronization

Implicit synchronization is available as only currently available elements are retrieved by snapshot push.

7.4.5 Self-description

Handshake is not available.

7.4.6 Communication/protocol

Communication features are implemented at PSM level. They are relevant for the specific platform chosen on which the EP will be implemented (e.g. http/XML, Web Services SOAP, REST, etc.).

7.4.7 General optimization issues

Some EP features of any context diagram features groups (e.g. information management, data delivery, etc.) allow the implementation of general optimization such as processing-saving and bandwith.

Payload timestamp information is available for client-side processing optimization made at the application level.

A push message may be generated for all clients reducing processing resources at the supplier side.

No extra optimization issues are considered in this EP+FEP.

8 Simple push

8.1 Overview

The "simple push" EP/FEP at PIM level is based on information messages sent or pushed by the supplier to a client. It can be implemented in several platforms: some examples are push of generated XML content by http/post, or a client providing a SOAP web service method “push” by which data can be provided by the supplier to the client.

This “simple push” adds extra features to the basic snapshot push EP to manage link monitoring, as well as synchronization/realignment in case of communication lacks or system maintenance. This mechanism will be detailed at PIM level in the following subclauses.

To describe the EP/FEP at PIM level all features are described in a general abstract format, independently from the specific technology platform in which this model will be implemented. (e.g. http/get XML, WebService).

Table 9 — Selection of features for simple push

Features area	Feature	Simple push available
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	N
	Link monitoring	Y
Information management	Operating modes	Periodic or On Occurrence based on triggering of supplier condition
	Update methods	Snapshot, Single Element Update, All Element Update
	Life cycle management	Y
Data delivery	Data delivery	Y
	Data request	N
	Large datasets handling	Y, optional
	Synchronization	Y, optional
Self-Description	Handshake	N
Communication	Security	At PSM level
	Compression	At PSM level
	Communication	At PSM level

8.2 Exchange pattern messages definition

8.2.1 Overall presentation

The information delivery business scenario description and definition state that data exchange is needed to align the information kept by the supplier system into the client system; for this purpose, an exchange system is used which provides tools enabling message generation and their transfer between supplier and client (see Figure 20).

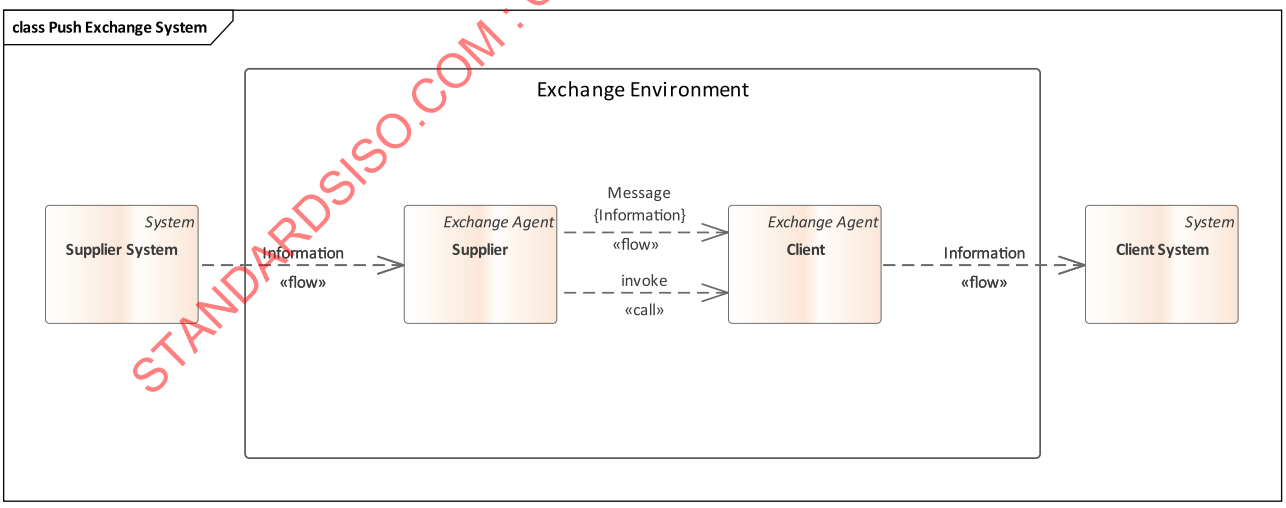

Figure 20 — Simple push exchange actors

The simple push EP is described in the following subclauses.

8.2.2 Basic exchange pattern

In a simple push context, the client provides a mechanism to receive data from an action taken at a supplier site, invoking specific resources / methods offered by the client.

Therefore, the supplier logically “pushes” messages to the client. The client shall acknowledge what is received by a return exchange information to the supplier. This exchange information return message is available to bring information back from the client to the supplier, such as SessionId, failure, success, snapshot synchronization request. Return message information is logically described in this PIM, while implementation will be defined at PSM level.

The simple push client provides two mechanism to the simple push supplier to push data:

— a "push" method is intended to push available data which had not yet been delivered to the client, based on some supplier side logic and status,

— a "snapshot push" is intended to push all currently available data, also called a snapshot of information, i.e. current information at a supplier system or last retrieved information for sampled data (see Annex F). This snapshot push method is used for synchronization purposes among client and supplier.

Besides these push data delivery methods, the simple push client also provides a keepAlive method to implement link monitoring capabilities among client and supplier. The keepAlive method is used from the supplier to advise the client when no information updates are to be delivered, so the supplier delivers a keepAlive message to check and enable the client to check that exchange systems and network connection are available, despite the supplier not needing to exchange payload content. KeepAlive messages are delivered by the supplier to the client, according to a time interval defined between them.

In the context of this simple push FEP+EP framework, to enable interoperability between client and supplier, all rules defined in this subclause apply.

Any simple push client exchange system shall realize a simple push client interface which provides a putSnapshotData, a putData and a keepAlive method.

Any simple push supplier exchange system shall realize a simple push supplier interface which can invoke the putSnapshotData, putData, keepAlive methods provided by the simple push client interface to deliver data or snapshot data and information to implement link monitoring.

Figure 21 shows the communication diagram for simple push FEP+EP.

In this FEP+EP framework the supplier pushes messages to the client.

The client shall acknowledge the received message by a return information to the supplier. This return information shall be coded as ExchangeInformation.

NOTE This return message is available to bring some exchange information from the client to the supplier which can be used for any further exchange feature implementations or application level checks or processing. This is out of scope of this document.

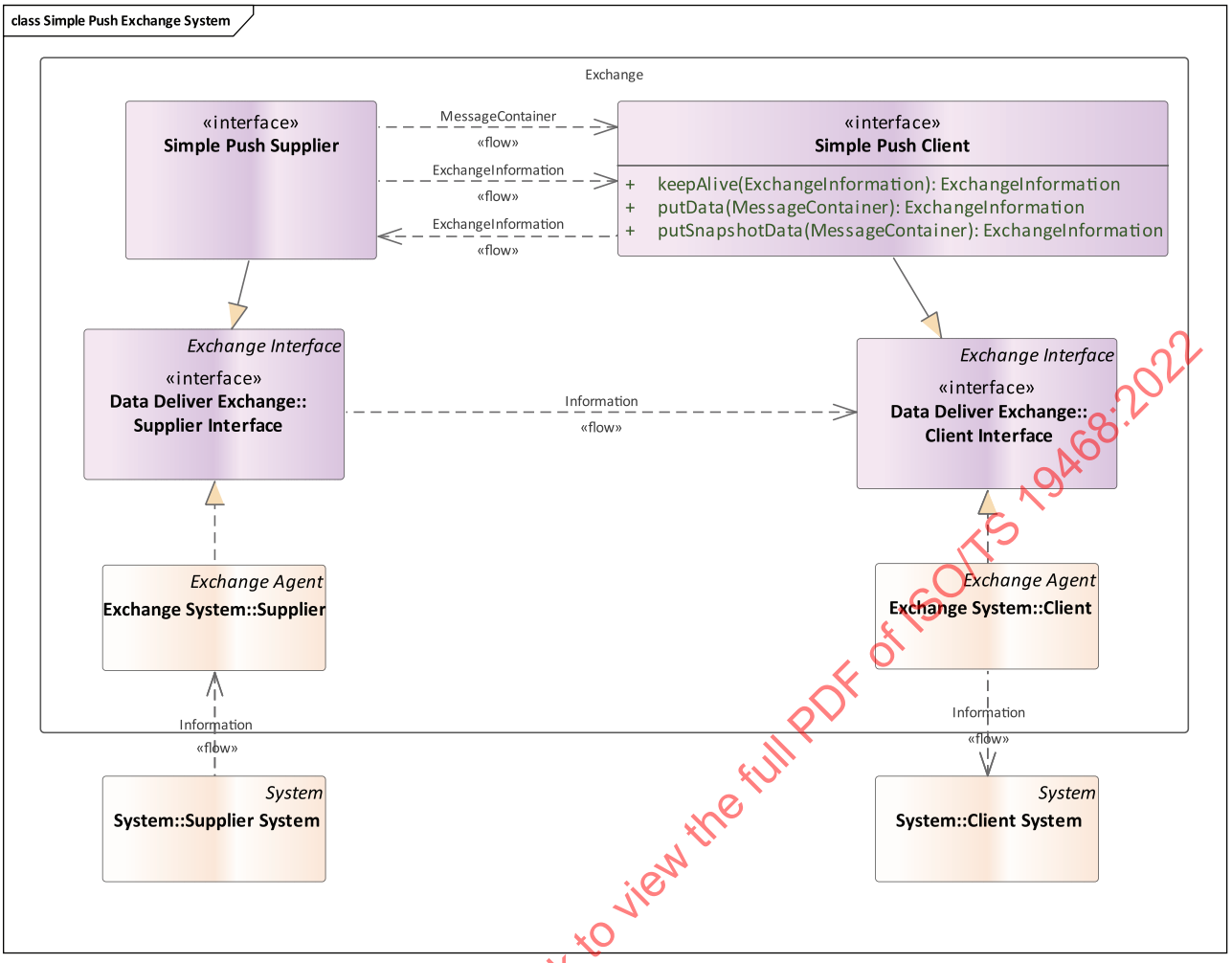

Figure 21 — Simple push exchange subsystems, interface interactions and methods

The supplier takes the initiative to push the data under the following conditions:

— **On occurrence push**: as soon as information is updated at the supplier systems, this condition triggers the supplier to send push data to the client for updating the client system as soon as possible after this update.

— **Periodic push**: at a predefined time interval, the supplier starts an exchange based on the client and supplier agreement (subscription contract).

— A **snapshot synchronization** with the whole currently available content snapshot at exchange initialization, e.g. the first-time data are exchanged among supplier and client.

— **Response to a snapshot synchronization request**: one snapshot alignment may also be transmitted to the client for internal system needs/maintenance/debug. It can be requested by the client via any return messages, i.e. as a specific return value in returned exchange information.

8.2.3 Relevant exchange information from exchange data model

8.2.3.1 Overview

A basic exchange data model has been provided to allow the implementation to deliver more payload contents on the same message and further information to allow managing extra features not required by the simple push exchange.

For interoperability convenience, the exchange data model wrapping shall be managed in this exchange.

A payload shall be pushed to a client using a basic exchange data model as reported in Figure 21.

An ExchangeInformation shall be returned from putData to convey information about exchange operation and connection status.

8.2.3.2 Exchange information

Information related to exchange that should be managed to make application development easier is fully described in the basic exchange data model.

Related exchange context information is:

— Supplier-related information

  — Requirement: supplier identification.

— Client-related information

  — Requirement: client identification.

Related dynamic information is:

— Exchange DynamicInformation (provided both by the client and the supplier) wraps information such as exchangeStatus ("Success", "Fail", "Close Session Request", "Snapshot Synchronization Request") and SessionID.

  — Requirement: session management, link monitoring.

— Message generation timestamp information.

  — Requirement: timely, reliable information, session management.

8.2.4 List of exchanged messages

Different messages or supplier/client interactions (invoked method) are exchanged in simple push which are needed to manage synchronization, payload exchange and link monitoring. These are formally contained in messages pushed to a client by a supplier or in messages returned from client to supplier. See Table 10.

Table 10 — List of messages types and detailed content

Interaction message	Direction supplier client	Designation	Description	Exchanged information	Optional
Payload push	Direct	putData	Push delivery of payload, which has to be delivered from supplier to client. Exchange information such as client and supplier identification and exchange status may be provided to easy controls.	Payload + Exchange information (MessageContainer)	N

Table 10 *(continued)*

Interaction message	Direction supplier client	Designation	Description	Exchanged information	Optional
Snapshot payload push	Direct	putSnapshotData	Push delivery of current available payload, i.e. snapshot after a first initialized session in case of first connection or after an explicit snapshot realignment request from the client. Exchange information such as client and supplier identification and exchange status may be provided to easy controls.	Snapshot payload + Exchange information (MessageContainer)	Y
KeepAlive	Direct	keepAlive	Test exchange link and confirm session validity when no payload push update is needed, exchange information such as client and supplier identification and exchange status may be provided to easy controls and supplier identification.	Exchange information	N
Exchange information return	Return	Exchange information	Exchange information is returned from client to supplier to provide return status i.e. success, fail, snapshot synchronization request and to easy controls such as supplier and client identification.	Exchange information	N

8.3 State diagrams

The supplier initiates the communication and is made aware of the client status based on the client return response to the supplier.

Figure 22 describes the client status as monitored and managed by the supplier.

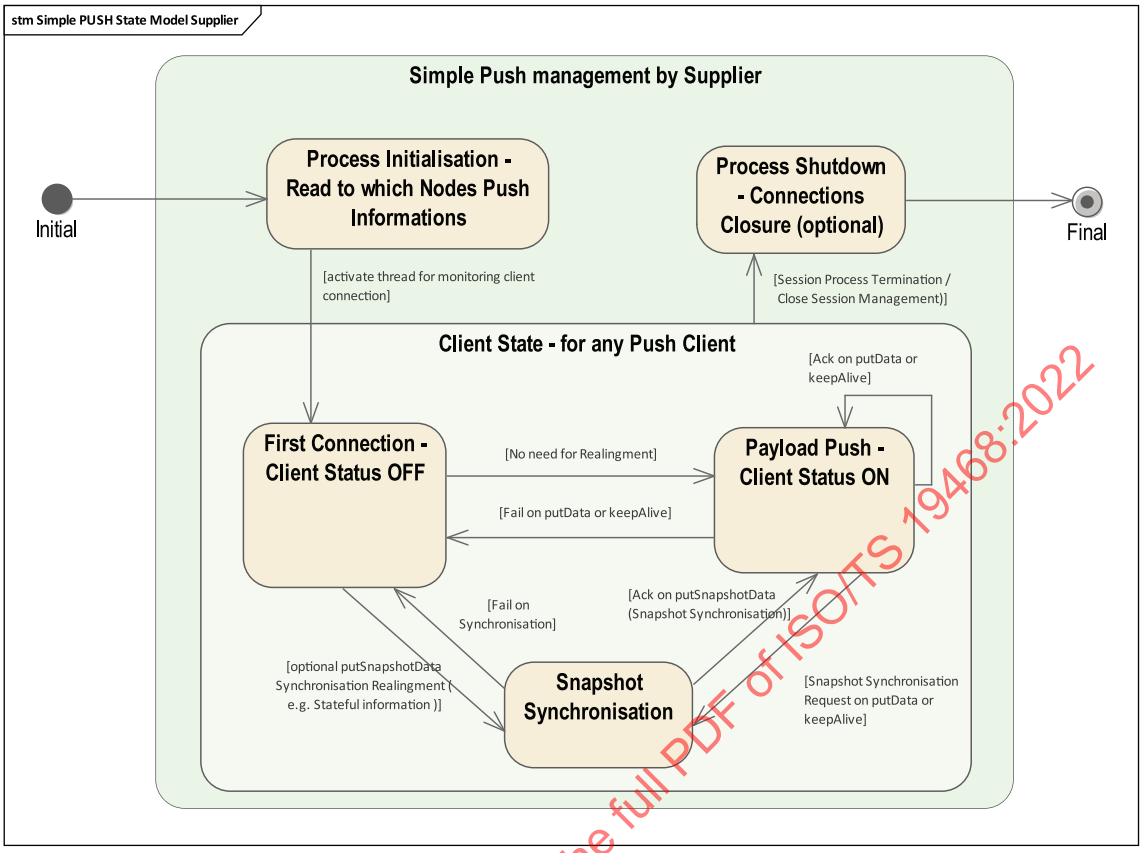

Figure 22 — Supplier-side simple push state diagram

Figure 23 describes the supplier status as monitored and managed by the client.

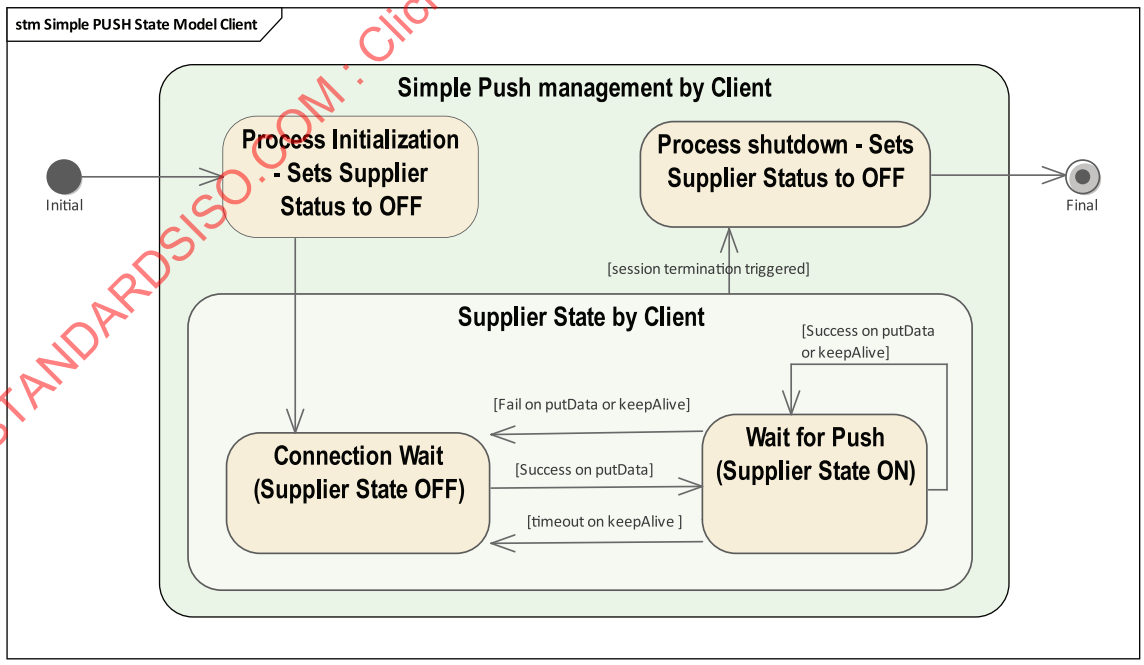

Figure 23 — Client-side simple push state diagram

Special management in initialization and termination of push process is to be considered at the application level in supplier and client systems.

8.4 Features implementation description

8.4.1 Overview

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the simple push data exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

8.4.2 Subscription contract

8.4.2.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in the client to assess the identity of supplier and authenticate the supplier request in messages exchange.

8.4.2.2 Catalogue

Managed offline, not automated.

8.4.3 Session

8.4.3.1 Session life cycle

No session is managed for the current EP+FEP.

8.4.3.2 Link monitoring

Link monitoring is done by payload push and keepAlive. When data is available, a payload push is exchanged which also informs the client and the supplier about the session and systems status: push received from the supplier on the client side and return of push on payload push on the supplier side guarantee the network is available and that the systems are up and running.

When no data is available at the supplier and a time-out has expired, a keepAlive message is exchanged to check network and system availability.

In case payload push or keepAlive fails or times out, the supplier assumes the client is offline and keeps track of its status for any management purposes at the supplier system side (delivery retry mechanism is to be described at PSM level defining a logical push mapping iterated for a maximum number of times). See Figure 24.

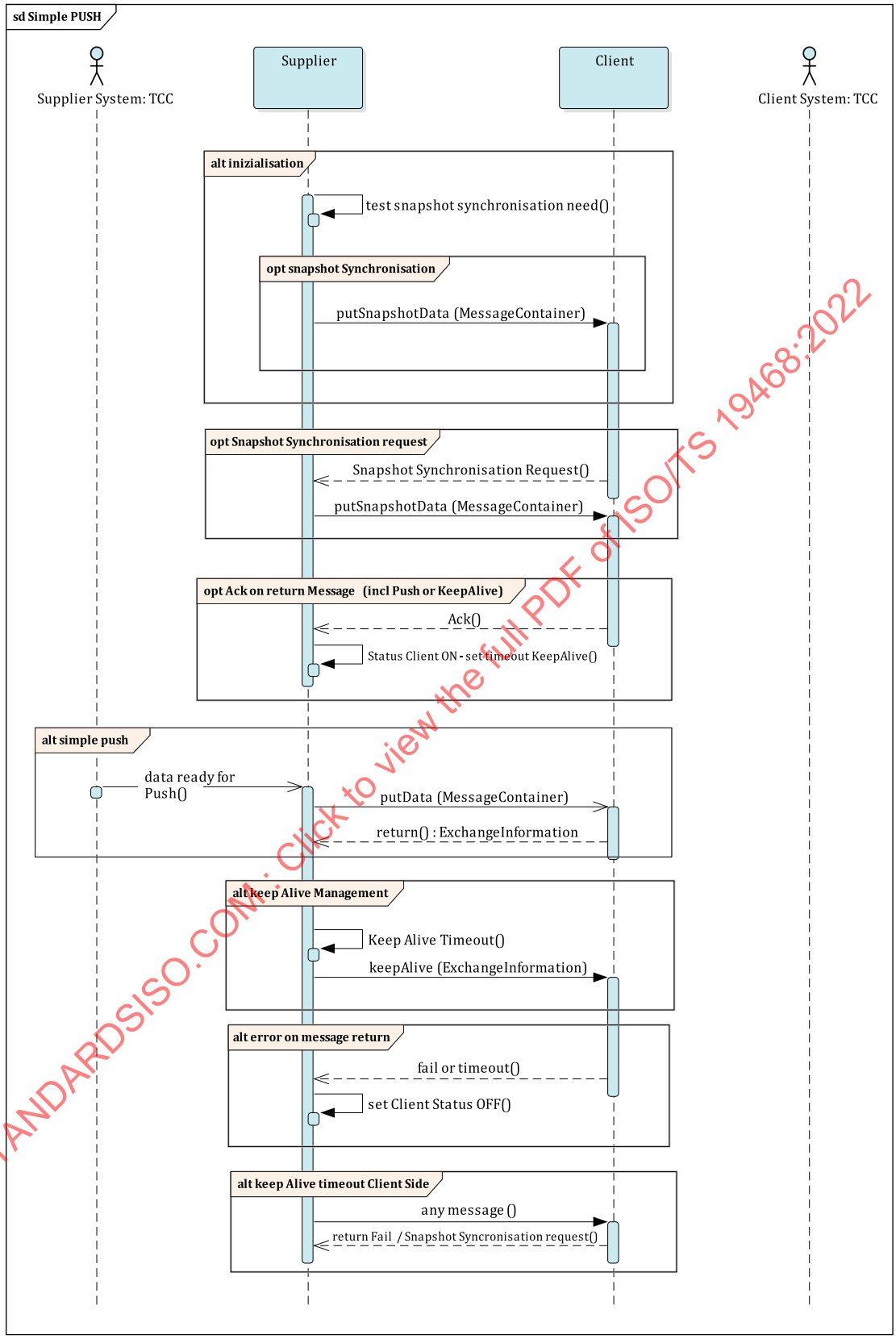

Figure 24 — Simple push sequence diagram for link monitoring and data delivery

8.4.4 Information management

8.4.4.1 Operating modes

Available operating mode for simple push is “Periodic”, or “On Occurrence” (i.e. condition-triggered based on supplier) push based on supplier-side conditions.

Description of operating modes is done at a general PIM level; no extra details are needed in this subclause for PIM/EP/FEP.

A payload push condition is triggered based on the agreed operating mode defined at subscription among client and supplier.

8.4.4.2 Update methods

Available updated methods are: snapshot, single element update, all element update.

All updates available are conveyed in a payload publication push message.

Description of update methods is done at PIM level; no extra details are needed in this subclause for PIM/exchange pattern/FEP.

8.4.4.3 Life cycle management

Description of life cycle management is done at PIM level.

Life cycle management to exchange data between client and supplier is embedded with the operating mode and update method chosen at subscription contract.

Push delivery method allows for conveying information from supplier to client as two different pieces of information.

Sampled data may be conveyed as Periodic or On Occurrence push of a snapshot payload containing all current active data or last collected data.

A "single element" or "all elements" update push can be done for any operating mode as well with On Occurrence or Periodic Push.

8.4.5 Data delivery

8.4.5.1 Data delivery

Based on the sequence diagram described in Figure 24, after initialization the supplier starts sending push information to a client. In case of stateful information delivery and based on the possibly agreed conditions of the subscription contract, for example, it manages a snapshot push whenever it has no historical information about client status, deriving it is the first connection ever and a snapshot push is needed to align the client system when it is not the first time the supplier sends to the client normal payload push data, but the client for internal system needs can also require for snapshot push data by its return status.

After initialization, ready data condition at the supplier system side triggers a payload push delivery.

A periodic push condition is also possible based on the contract agreement between supplier and client.

See sequence diagram at link monitoring life cycle for all messaging details.

8.4.5.2 Data request

Not fully implemented in this pattern.

A snapshot synchronization request can be managed in the client return data, by the return status described in the basic exchange data model by returnStatus set as snapshotSynchronizationRequest.

This feature implementation is not mandatory in this FEP. Agreement to manage between client and supplier is needed to enable full interoperability.

8.4.5.3 Large datasets handling

Not fully implemented in this pattern.

A multi-part data delivery can be optionally implemented by the supplier by setting the completedPayload as false in the delivered message within the exchange dynamic information setting, as described in the basic exchange data model. It will inform the client that one or more subsequent message will be delivered to complete the payload information, until the attribute completedPayload is set as true.

This feature implementation is not mandatory in this FEP. Agreement to manage among client and supplier are needed to enable full interoperability.

8.4.5.4 Synchronization

Snapshot synchronization and delta synchronization are available in simple push.

Snapshot synchronization is optionally managed at first connection with a client or under client request. In all other cases a simple push is delivered based on supplier site data available and conditions.

8.4.6 Self-description

Handshake not available.

8.4.7 Communication/protocol

Communication features are implemented at PSM level. They are relevant to the specific platform chosen on which the EP will be implemented (e.g. http/XML, Web Services SOAP, REST).

8.4.8 General optimization issues

Some EP features of any context diagrams features groups (e.g. information management, data delivery, etc.) allow the implementation of general optimization such as processing saving and bandwith.

Payload timestamp information is available for client-side processing optimization made at the application level.

Snapshot push messages may be generated for all clients reducing processing resources at the supplier-side.

No extra optimization issues are considered in this EP+FEP.

9 Stateful push

9.1 Overview

The "stateful push" EP/FEP at PIM level is based on information messages sent or pushed by a supplier to a client. This exchange pattern can be implemented in several platforms: some examples are pushing generated XML content by http/post, or client providing a SOAP Web service method “push” by which data can be provided to a client by a supplier.

This “stateful push” adds extra features to the snapshot push EP in order to manage session life cycle and link monitoring, as well as synchronization/realignment in case of communication breaks or system

maintenance. Further for data delivery features it enables, besides snapshot payload delivery, any update of information e.g. allowing single element updates. These mechanisms will be fully explained at the PIM level in the following subclauses.

To describe the EP/FEP at PIM level, all the features are described in a general abstract format, independently from the specific technologic platform in which this model will be implemented (e.g. http/get XML or Web services). See Table 11.

Table 11 — Selection of features for stateful push

Features area	Feature	Stateful push available
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	Y
	Link monitoring	Y
Information management	Operating modes	Periodic or On Occurrence based on supplier triggering conditions
	Update methods	Snapshot, Single Element Update, All Element Update
	Life cycle management	Y
Data delivery	Data delivery	Y
	Data request	Snapshot realignment
	Large datasets handling	Y, optional
	Synchronization	Y
Self-description	Handshake	N
Communication	Security	At PSM level
	Compression	At PSM level
	Communication	At PSM level

9.2 Exchange pattern messages definition

9.2.1 Overall presentation

The information delivery business scenario description and definition state that data exchange is needed to align the information kept by the supplier system into the client system; for this purpose, an exchange system is used which provides tools enabling messages generation and their transfer between a supplier and a client (see Figure 25).

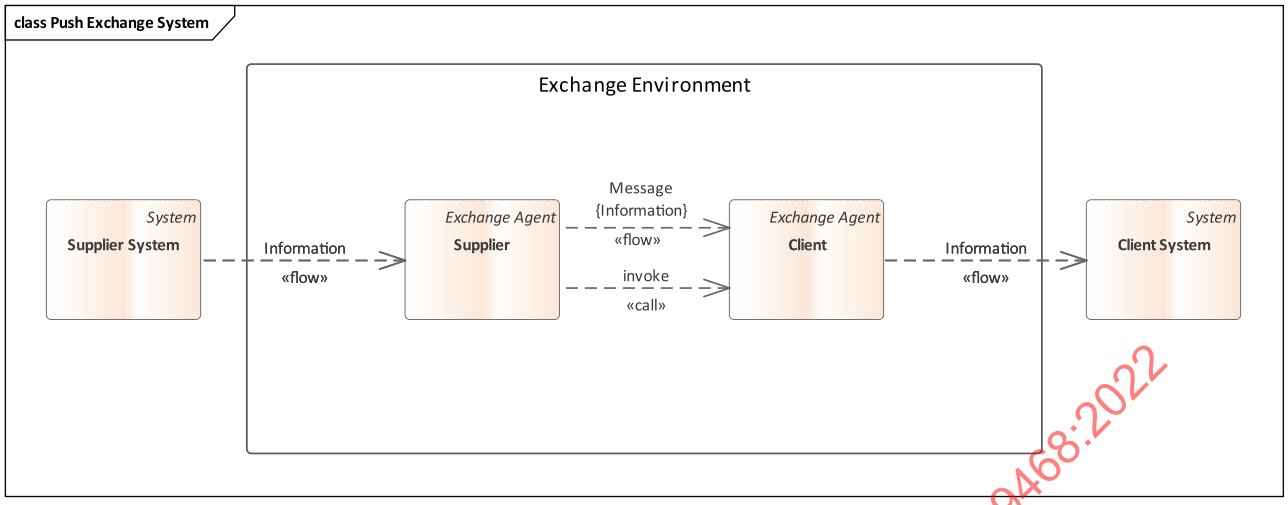

Figure 25 — Stateful push exchange actors

The stateful push exchange pattern is described in the following subclauses.

9.2.2 Basic exchange pattern

In a stateful push context, the client provides a mechanism to receive data from an action taken at the supplier site invoking specific resources / methods offered by the client.

Therefore, the supplier logically “pushes” messages to the client. The client shall acknowledge what is received by a return exchange information to the supplier. This exchange information return message is available to bring information back from the client to the supplier, such as SessionId, failure, success or snapshot synchronization request. Return message information is logically described in this PIM, while implementation will be defined at PSM level.

As in simple push, the stateful push client provides two mechanism to the stateful push supplier to push data:

— a "push" method is intended to push available data which has not yet been delivered to the client, based on supplier-side logic and status,

— a "snapshot push" is intended to push all currently available data. This is also called a snapshot of information, i.e. current information at supplier system or last retrieved information for sampled data (see Annex F). This snapshot push method is used for synchronization purposes between client and supplier.

Besides push data delivery methods, the simple push client also provides a keepAlive method to implement link monitoring capabilities among client and supplier. The keepAlive method is used from the supplier to advise the client when no information updates are to be delivered, so the supplier delivers a keepAlive message to check and enable the client to check that exchange systems and network connection are available, despite the supplier not needing to exchange payload content. KeepAlive messages are delivered by the supplier to the client, according to a time interval which is defined between them.

Stateful push session management methods are available to implement session management features. Such methods include namely openSession and closeSession usage and are used to define dynamic exchange information context to enable session management exchange features.

In the context of this "Stateful Push" FEP+EP framework, to enable interoperability among client and supplier, all rules defined in this subclause apply.

Any stateful push client exchange system shall realize a stateful push client interface which provides a putSnapshotData, a putData, an openSession, a closeSession method and a keepAlive method.

Any stateful push supplier exchange system shall realize a stateful push supplier interface which can invoke the putSnapshotData, putData, openSession, closeSession and keepAlive methods provided by the stateful push client interface to deliver data or snapshot data and information to implement link monitoring and session management.

Figure 26 shows the communication diagram for simple push FEP + EP.

In this FEP+EP framework the supplier "pushes" messages to the client.

The client shall acknowledge the received message by a return information to the supplier. This return information shall be coded as ExchangeInformation.

NOTE This return message is available to bring some exchange information from the client to the supplier which can be used for any further exchange feature implementations or application level checks or processing. This is out of scope of this document.

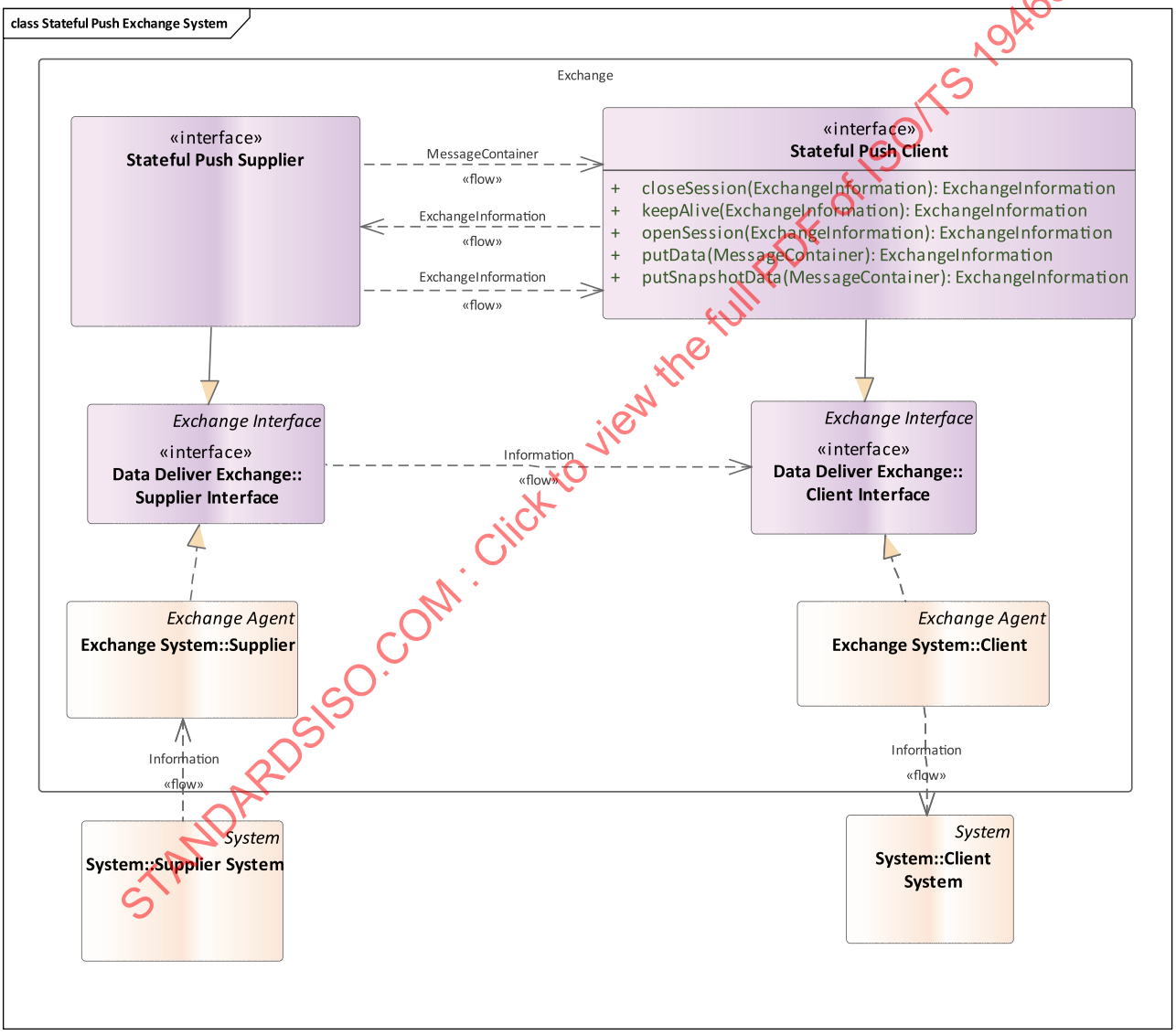

Figure 26 — Stateful push exchange subsystems, interface interactions and methods

The supplier takes the initiative to push the data, or the snapshot data, under the following conditions:

— **On occurrence push**: as soon as an information is updated at the supplier systems, this condition triggers the exchange supplier to manage an alignment to the exchange client to update the client system as soon as possible after this update.

- **Periodic push**: at predefined time interval, the supplier starts an exchange based on hte client and supplier agreement (subscription contract).
- A **snapshot synchronization** with the whole currently available content snapshot at exchange initialization, e.g. the first-time data are exchanged among supplier and client.
  - **First session initialization:** a global snapshot alignment is needed to convey all currently valid data at the first connection of exchange.
  - **Session initialization**: after a session has been created, the client shall be aligned with an incremental delta synchronization when a partial update feature is enabled, delivering all updated content since last exchange, or with a global synchronization with all the current active content in case the snapshot alignment feature is enabled (this may also depend on specific payload depending on the agreement between client and supplier or contract).
- **Response to a snapshot synchronization request**: one snapshot alignment may also be transmitted to the client for internal system needs/maintenance/debug. It can be requested by the client via any return messages, i.e. as a specific return value in returned exchange information**.**

9.2.3 Relevant exchange information from exchange data model

A basic exchange data model has been provided to allow an implementation that delivers more payload contents in the same message and further information to allow managing extra features which are not required by the basic snapshot push exchange.

In order to ensure interoperability, the exchange data model wrapping shall be used in this exchange pattern.

A container shall be pushed to a client using a basic exchange data model as stated in Figure 26.

An ExchangeInformation object shall be returned from putData to convey information about exchange operation and connection status.

9.2.3.1 Exchange information

Information related to exchange that should be managed to make application development easier is fully described in the basic exchange data model.

Related exchange context information is:

- Supplier-related information
  - Requirement: supplier identification.
- Client-related information
  - Requirement: client identification.

Related dynamic information is:

- Exchange DynamicInformation (provided both by the client and the supplier) wraps information such as exchangeStatus ("Success", "Fail", "Close Session Request", "Snapshot Synchronization Request") and SessionID.
  - Requirement: Session management, Link monitoring.
- Message generation timestamp information
  - Requirement: timely, reliable Information, session management.

9.2.3.2 Payload information

— Generation timestamp information

  — Requirement: timely, reliable Information, session management.

9.2.4 List of exchanged messages

Different messages or supplier/client interactions are exchanged in the Stateful Push which are needed to manage session, synchronization, payload exchange, link monitoring. These are formally contained in messages pushed to a client by a supplier or in messages returned from client to supplier. See Table 12.

Table 12 — List of message types and detailed content

Interaction message	Direction supplier client	Designation	Description	Exchanged information	Optional
Open session	Direct	openSession	The supplier initializes a push delivery session.	Exchange information	N
Payload push	Direct	putData	Push delivery of payload, which has not been yet delivered from supplier to client. It shall contain in exchange information the session ID, previously obtained with OpenSession, referring to the session for which it is pushing data.	Payload + Exchange information (MessageContainer)	N
Snapshot payload push	Direct	putSnapshotdata	Push delivery of current available payload, i.e. snapshot after a first initialized session in case of first connection or after an explicit snapshot realignment request from the client. It shall contain in exchange information the session ID, previously obtained with OpenSession, referring to the session for which it is pushing data.	Snapshot Payload + Exchange information (MessageContainer)	N
KeepAlive	Direct	keepAlive	Test exchange link and confirm session validity when no payload push update is needed. It shall deliver the Session ID of a previously opened session, wrapped in exchange information.	Exchange information	N
Close session	Direct	closeSession	Message to gracefully close a delivery session, initiated by the supplier	Exchange Information	N
Exchange information return	Return	D2Exchange Infomation	Exchange information is returned from client to supplier to provide return status i.e. success, fail, snapshot synchronization request and to easy controls such as supplier and client identification.	Exchange information	N

9.3 State Diagrams

The supplier initiates the communication and can be aware of the client status based on the client return response to the supplier.

Figure 27 describes the client status as monitored and managed by the supplier.

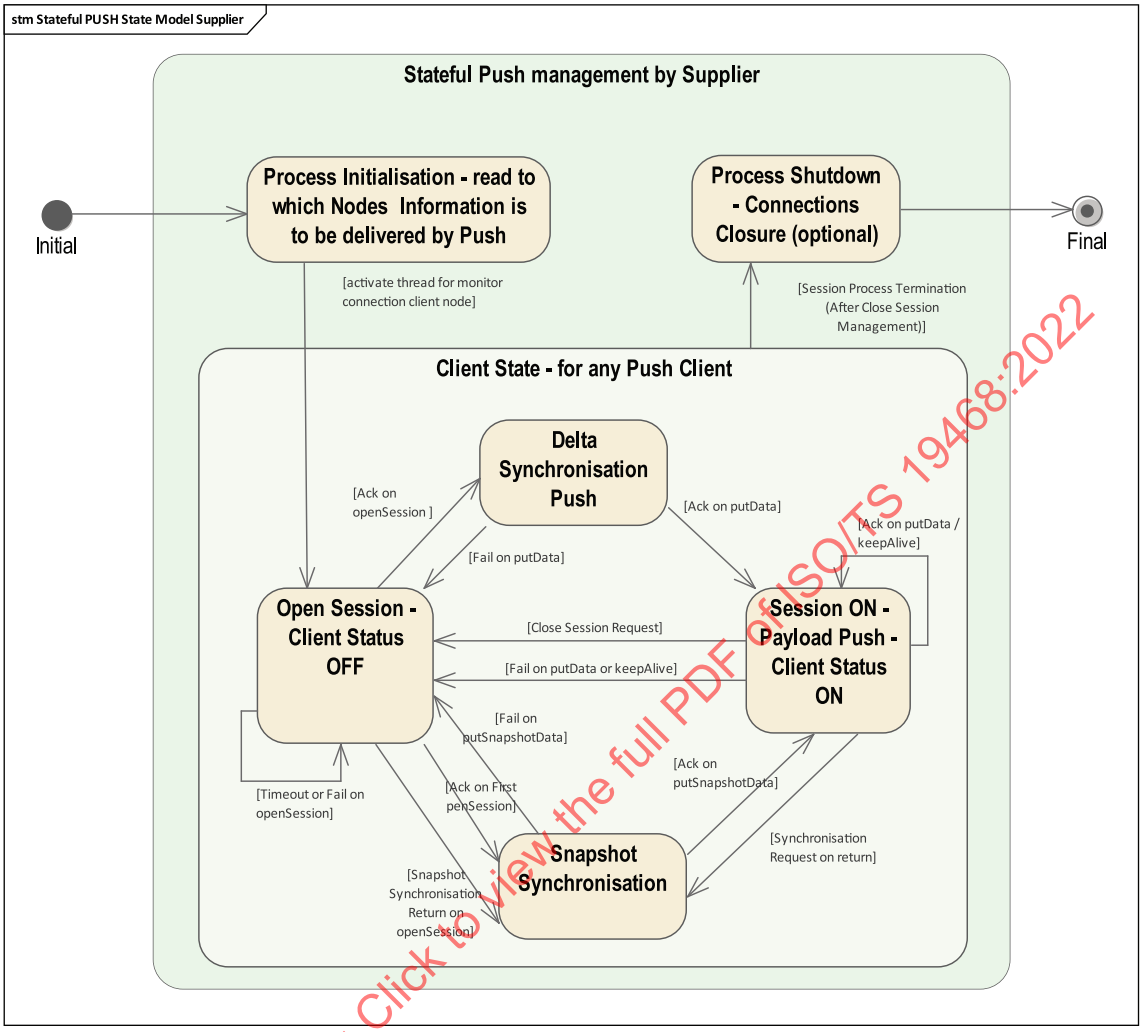

Figure 27 — Supplier-side Stateful Push state diagram

Figure 28 describes the supplier status as monitored and managed by the client.

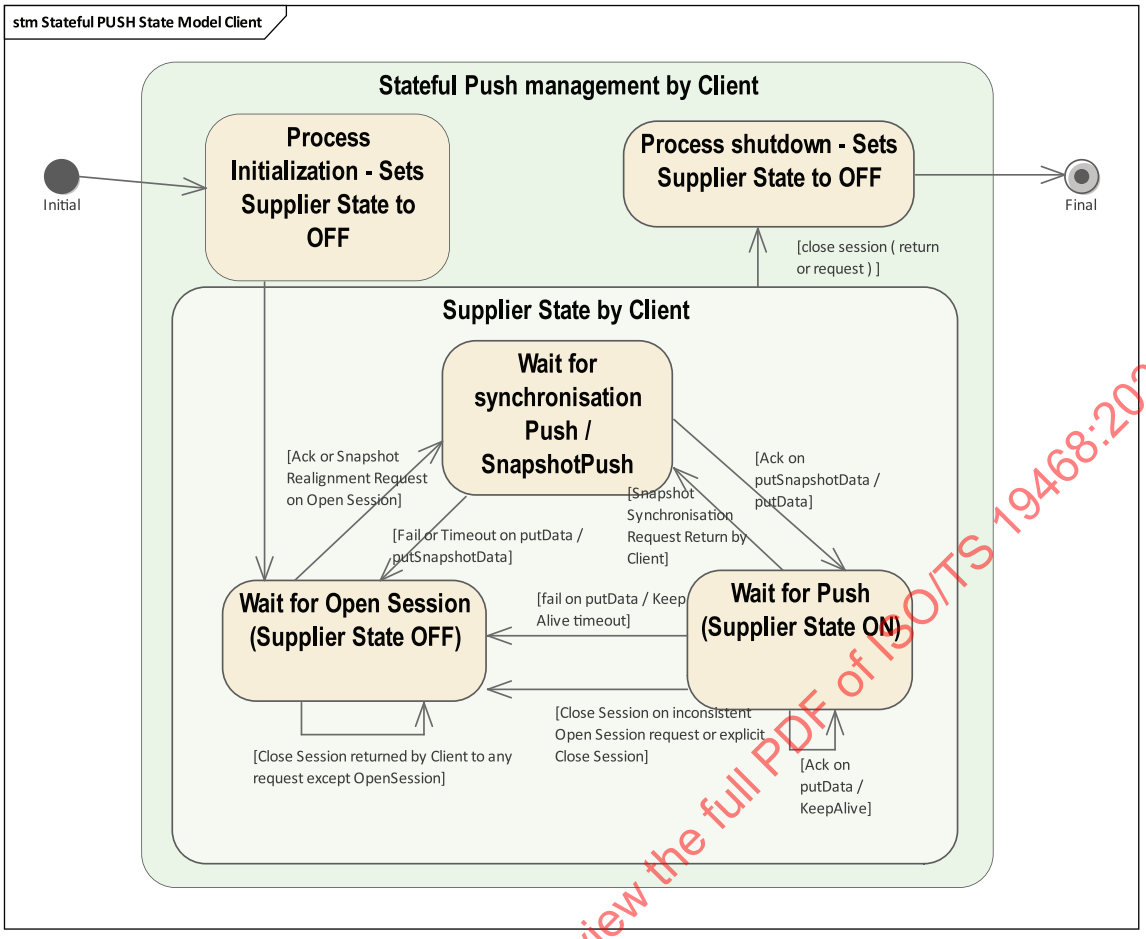

Figure 28 — Client-side Stateful Push state diagram

Specific management in the initialization and termination of a push process should be considered at the application level in the supplier and client systems.

9.4 Features implementation description

9.4.1 Overview

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the publish-subscribe data exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

The corresponding classes, attributes and relationships described in the class diagrams included in the next subclauses are described in C.4.

9.4.2 Subscription contract

9.4.2.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in the client to assess the identity of supplier and authenticate the supplier request in messages exchange.

9.4.2.2 Catalogue

Managed offline, not automated.

9.4.3 Session

9.4.3.1 Session life cycle

After the session status management diagrams, the following sequence diagram illustrates the exchanged message and the expected return and behaviour.

When a session needs to be initiated, an “Open Session” is tried in loop until it succeeds.

A failure when opening a session includes cases of a client who has not subscribed or is not authorized. Checking this can be ensured at the PSM level (e.g. this could include VPN setting or IP firewall or signatures handling), logical information may be included in exchange data to be managed in a subscription check at the client side.

When a session is "on" the supplier pushes available payload data to the client in case one of the following two conditions is fulfilled:

— payload available for On Occurrence operating mode, or

— payload delivery time-out for Periodic push operating mode.

In case no payload, is available a keepAlive message is used to check session status for the supplier and the client.

When no keepAlive message or no payload is received, after a keepAlive time-out, the client invalidates the session on its side and returns a "close session" message to prevent any attempt of push from the supplier.

If any push or keepAlive fails, the supplier invalidates the session and starts a new loop to open a session.

Realignment messages are managed when a session is opened; a global synchronization request from the client is returned in opening session when needed. Any global synchronization may be requested by a client in any push return as well, but this does not close the session.

Any message and return in the sequence diagram (Figure 29) will be mapped in PSM definition to real platform available implementation such as a web service "service request and return" or any other available mechanism in a specific platform.

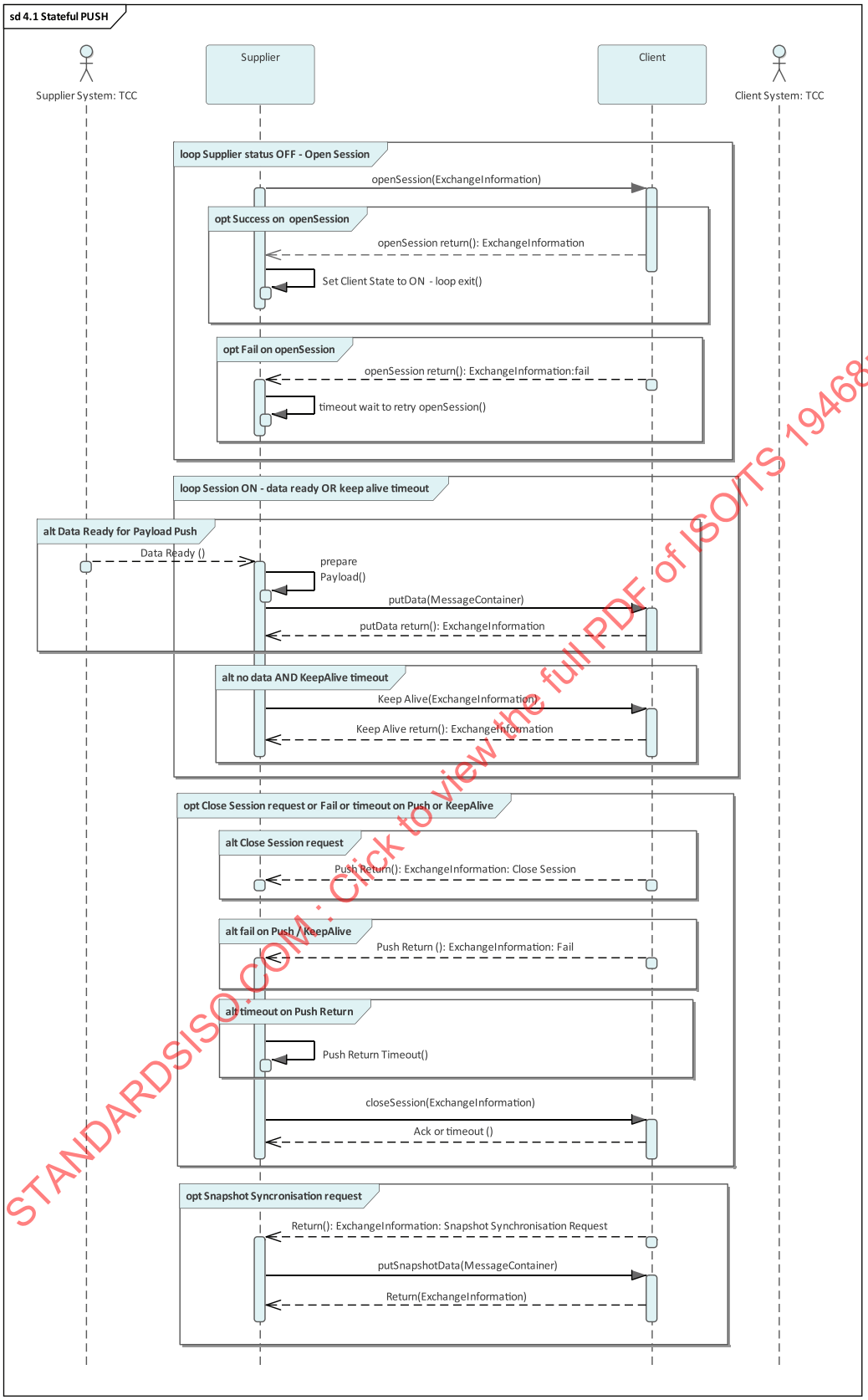

Figure 29 — Stateful push sequence diagram for session life cycle and data delivery

9.4.3.2 Link monitoring

KeepAlive is based as described in the previous session life cycle.

When data is available, a payload push is exchanged which informs the client and the supplier about the session and systems status. The push is received from the supplier on the client side and the return of push on payload push on the supplier side guarantees the network is available and the systems are up and running.

When no data is available and a time-out has expired, a keepAlive message is exchanged to check the network and system availability.

In case a payload push or a keepAlive fails, the session shall be invalidated (the retry mechanism is described at the PSM level introducing a logical push mapping iterated for a maximum number of attempts).

9.4.4 Information management

9.4.4.1 Operating modes

The available operating modes for supplier push are either periodic, or condition-triggered (based on the supplier-side conditions and the interchange agreement at the subscription).

The general description of the operating modes is done at the PIM level. No extra details are needed in this subclause for PIM/EP/FEP.

A payload push is triggered based on the agreed operating mode defined at the subscription between the client and the supplier.

9.4.4.2 Update methods

Available updated methods are: snapshot, single element update, all elements update.

All updates available are conveyed in a payload publication push message.

The general description of the update methods is done at the PIM level, no extra details are needed in this subclause for PIM/exchange pattern/FEP.

9.4.4.3 Life cycle management

The description of life cycle management is done at the PIM level.

The life cycle management for exchanging data among a client and a supplier is embedded in the operating mode and the update method which have been chosen in the subscription contract.

The push delivery method allows conveying information from a supplier to a client as two different sets of information.

Sampled data can be conveyed (pushed) as “Periodic” or “On occurrence” of a global payload containing all currently active data or last collected data.

A "single element" or "all elements" update push can be done for any operating mode, i.e. with "On occurrence" or "Periodic" push.

9.4.5 Data delivery

9.4.5.1 Data delivery

Session life cycle online section allows sending data when available at the supplier system by triggering a data ready condition.

A periodic push condition is also possible based on contract agreement among supplier and client.

See sequence diagram at the session life cycle for messaging details.

9.4.5.2 Data request

Not fully implemented in this pattern.

A snapshot synchronization request can be managed in the client return data, by the return status described in the basic exchange data model by returnStatus set as snapshotSynchronizationRequest.

This feature implementation is not mandatory in this FEP, agreement to manage among client and supplier are needed to enable full interoperability.

9.4.5.3 Large datasets handling

Not fully implemented in this pattern.

A multi-part data delivery can be optionally implemented by the supplier by setting the completedPayload as false in the delivered message within exchange dynamic information setting, as described in the basic exchange data model. It will inform the client that one or more subsequent messages will be delivered to complete the payload information, until the attribute completedPayload will be set as true.

This feature implementation is not mandatory in this FEP. Agreement to manage among client and supplier are needed to enable full interoperability.

9.4.5.4 Synchronization

Global synchronization and delta synchronization are available in stateful push.

The global synchronization is managed at the first session with a client or under a client request.

The delta synchronization is managed once a session has been created and closed due to a network error or any other condition, so that not-delivered payload data is stored at the supplier side and delivered to the client as soon as a session is established again.

9.4.6 Self-description

The handshake is not available.

9.4.7 Communication

The communication features are implemented at the PSM level. They are relevant to a specific platform chosen on which the exchange pattern will be implemented (e.g. http/XML, Web services with SOAP, REST, etc.).

9.4.8 General optimization issues

Some EP features of any context diagrams features groups (e.g. information management, data delivery, etc.) allow the implementation of general optimization such as processing saving and bandwith.

Payload timestamp information is available for client-side processing optimization made at the application level.

Snapshot push messages may be generated for all clients reducing processing resources at the supplier-side.

No extra optimization issues are considered in this EP+FEP.

10 Simple CIS

10.1 Overview

Simple CIS FEP+EP PIM is based on a description of interactions which enable service request and feedback exchange among a service requester and one to many service providers as illustrated in Annex G.

It can be implemented in several technological platforms with specific interactions methods, e.g. SOAP, WebService methods.

The simple CIS FEP+EP framework enables a common communication interface to embed ITS service requests and feedback in a way that enables two or more TMC, TIC or SP systems to perform in a common, interoperable way.

The simple CIS FEP+EP framework is not designed to implement data delivery business scenarios, but it may be based on payload content which is assumed to be exchanged by data delivery which can be enabled by data delivery FEP+EP. Only specific features enabling CIS are described and introduced for simple CIS. Features which are not related to this FEP+EP are marked as "Not applicable" in Table 9.

To describe the EP+FEP at PIM leve,l all features are described in a general abstract format, independently from the specific technology platform in which this model will be implemented. (e.g. http/get XML, WebService). See Table 13.

Table 13 — Selection of features for simple CIS

Features area	Feature	Simple push available
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	N
	Link monitoring	N
Information management	Operating modes	Not applicable
	Update methods	Not applicable
	Life cycle management	Not applicable
	Support information processing	Y
	Distributed transaction	Y, not atomic transactions
Data delivery	Data delivery	Not applicable
	Data request	Not applicable
	Large datasets handling	Not applicable
	Synchronization	Not applicable
Self-description	Handshake	N
Communication	Security	At PSM level
	Compression	At PSM level
	Communication	At PSM level

10.2 Exchange pattern and messages definition

10.2.1 Overall presentation

Simple CIS FEP+EP enables one service requester node to implement CIS service request interaction for one to several service providers.

The interaction is described as the CIS service requester addressing all involved CIS service providers through their simple CIS exchange interfaces, which provides methods for delivering service requests from a requester to a multiplicity of service providers (Figure 30).

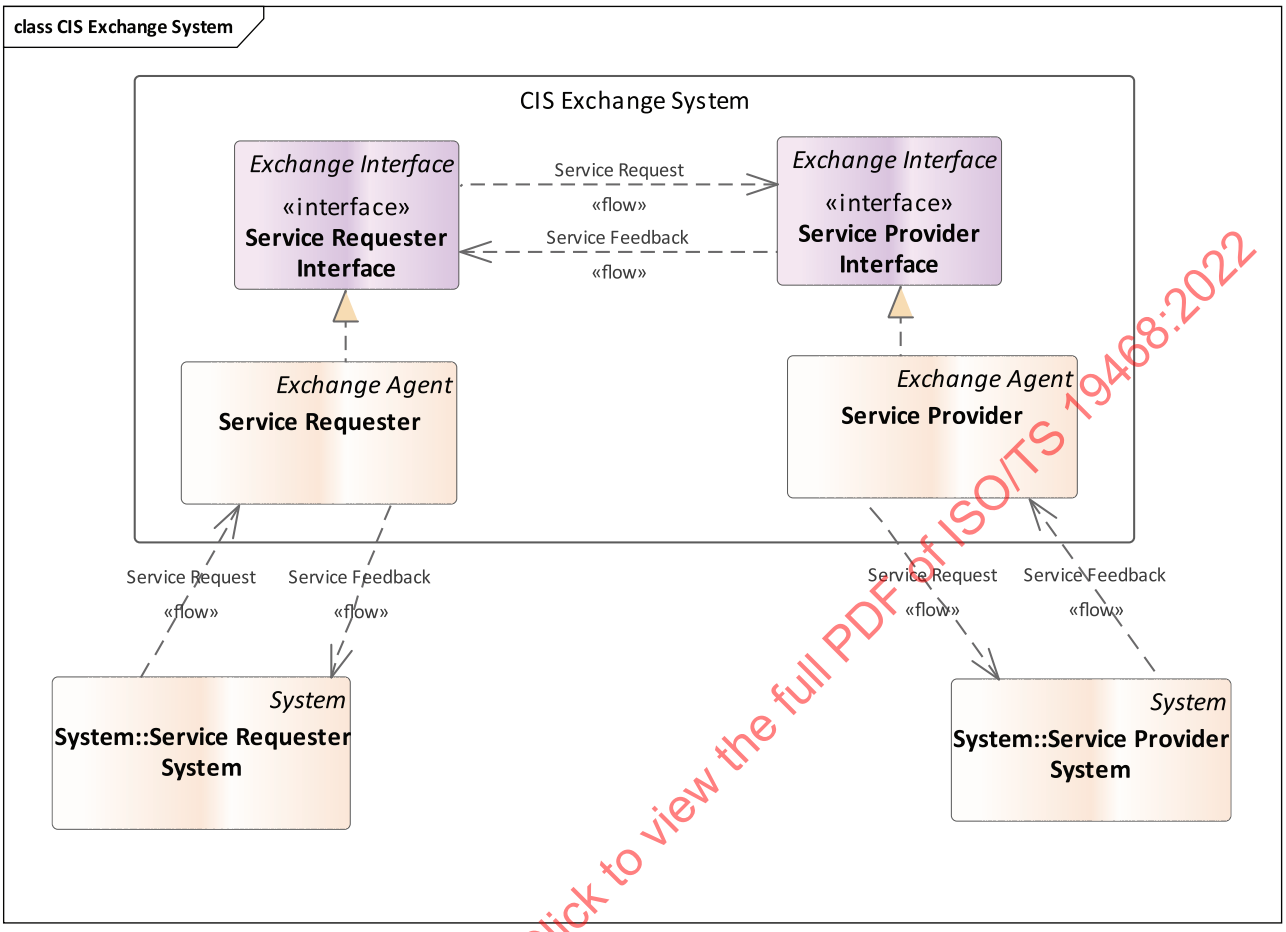

Figure 30 — Simple CIS exchange actors

The simple CIS EP described in the following subclauses.

10.2.2 Basic exchange pattern

In a simple CIS context the service provider provides a mechanism to receive a service request from action taken at a service requester site. This will result in the service requester invoking specific methods/resources offered by the service provider. On the reverse side, a mechanism is also needed to enable the service provider to feed back the result of processing or action triggered by the service request, so the service requester itself provides a mechanism to receive a service feedback from the involved service providers when their actions or processes results are available.

Figure 31 shows the communication diagram for simple CIS FEP + EP.

In this FEP+EP framework, the service requester delivers a service request trigger to the service provider and the service provider delivers service feedback information to the service requester.

Therefore, the service requester logically "pushes" CIS service requests messages, embedded in a MessageContainer, to the service provider, which shall acknowledge the reception of such messages by a return message, embedded in an ExchangeInformation. This exchange information return message is available to bring information back from the service provider to the service requester, such as failure, success. Return message information is logically described in this PIM, while implementation will be defined at PSM level.

At the same time, the involved service providers logically push CIS service feedback messages, embeded in a MessageContainer to the service requester, which symmetrically shall acknowledge the reception of suche messages by a return message, embedded in an ExchangeInformation. This exchange information return message is available to bring some information back to the service provider, the management of which is not relevant to this FEP+EP and is not described in this document.

In this workflow framework pattern, the management of any further workflow based on any processing or action errors is based on CIS service feedback. It is only in charge of the service requester system and it is defined at application level based on the specific application requirements needed to enable the specific collaborative ITS services.

In the context of this simple push FEP+EP framework, to enable interoperability among the CIS service requester and CIS service providers, all rules defined in this subclause apply.

Any simple CIS service provider exchange system shall realize a simple CIS service provider interface which provides a putCISServiceRequest method.

Any simple push service requester exchange system shall realize a simple CIS service requester interface which can invoke the putCISServiceRequest method provided by the simple CIS service provider interface to deliver the CIS service request to trigger action/process by the service provider systems.

Any simple CIS service requester exchange system shall realize a simple CIS service requester interface which provides a putCISServiceFeedback method.

Any simple CIS service provider exchange system shall realize a simple CIS service provider interface which can invoke the putServiceRequest method provided by the simple CIS service requester interface to deliver CIS service feedback to the service requester systems.

Figure 31 shows the communication diagram for simple push FEP+EP.

In this FEP+EP framework the service requester pushes service request messages to the service provider and the service provider pushes service feedback messages to the service requester.

Both service provider and service requester shall acknowledge the received service request or service feedback messages by a return information to the corresponding counterpart, i.e. respectively to the service requester and to the service provider. This return information shall be coded as ExchangeInformation.

NOTE This return message is available to bring some exchange information back from the receiver system which can be used for any further exchange features implementations or application level checks or processing and/or workflow management decision in CIS implementation. This is out of scope of this document.

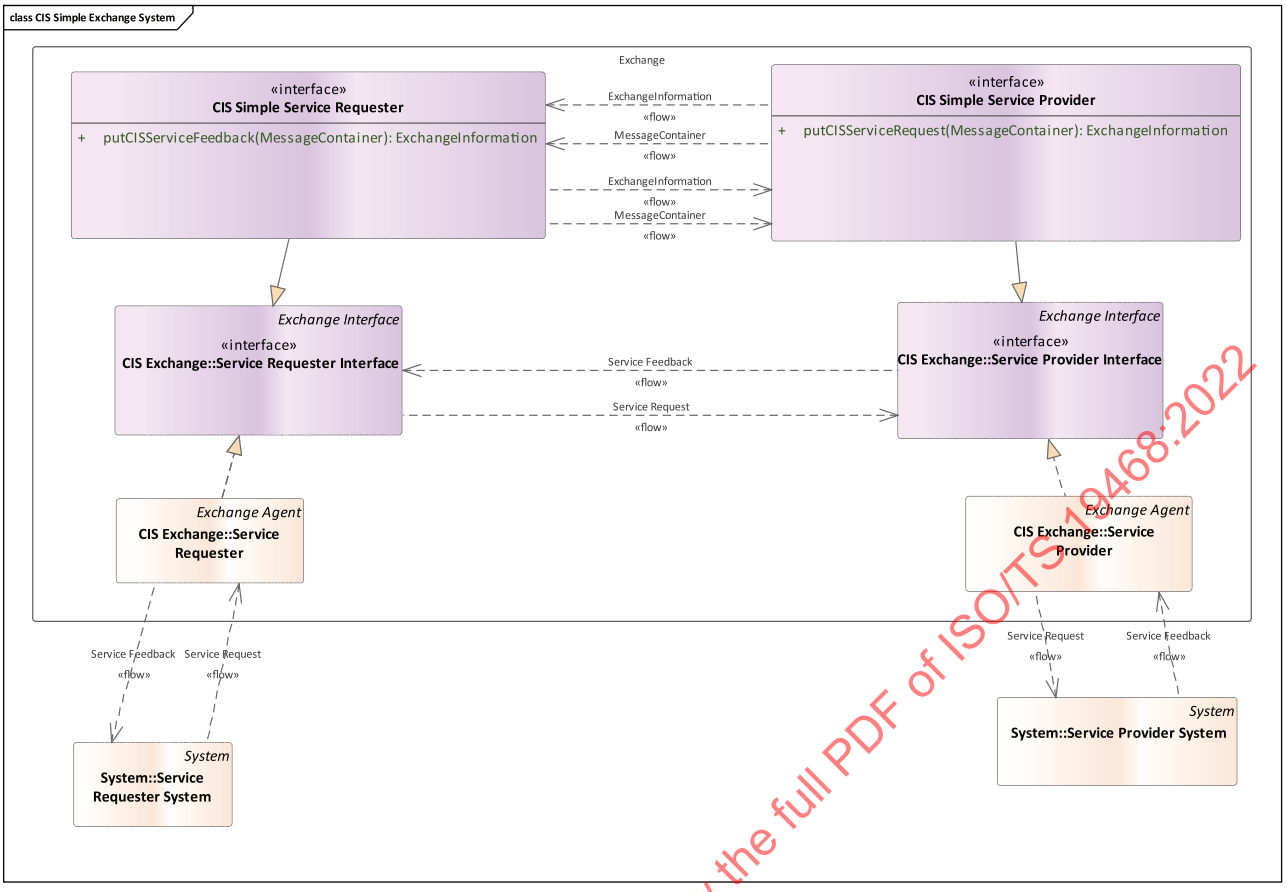

Figure 31 — Simple CIS exchange subsystems, interface interactions and methods

10.2.3 Relevant exchange information from exchange data model

10.2.3.1 Overview

A basic exchange data model has been provided to allow the implementation to enable CIS features to support information processing and enable distributed transactions features.

CIS information embeds all information needed to request for specific processing and to enable workflow management to support transaction negotiation processes by service request and service feedback messages.

Exchange information enables delivery of the exchange context of the involved service requester and service providers to enable orchestration of ITS services supplying and exchanging dynamic information which enables general exchange status information to monitor transactions negotiation or service processing in general.

10.2.3.2 Exchange information

Some non-mandatory information which should be managed in the exchange information for easy application development is fully described in the basic exchange data model:

— Service provider identification

  — Requirement: supplier identification. (String)

— Client identification

  — Requirement: client identification. (String)

— Exchange information (provided both by the client and the supplier) wraps exchange and exchange status information "Success", "Fail", "Close Session Request", “SessionId”.

— Generation timestamp information

  — Requirement: timely, reliable information.

10.2.3.3 CIS information

— Service request

  This provides the reference to elements to be processed and the requested action, also addressing the service for which the action is triggered.

— Predefined service

  — Requirements: support for information processing, distributed transaction.

— Requested action

  — Requirements: support for information processing, distributed transaction, distributed atomic transaction.

Service Feedback

  — Requirements: support for feedback on information processing, distributed transaction.

Service request status

  — Requirements: support information processing, support for feedback on information processing, distributed transaction.

10.2.4 Exchanged messages

Different messages or supplier/client interactions (invoked method) are exchanged in simple push which are needed to manage synchronization, payload exchange and link monitoring. These are formally contained in pushed messages to a client from a supplier or in return messages from client to supplier. See Table 14.

Table 14 — List of messages types and detailed content

Interaction message	Direction requester provider	Designation	Description	Exchanged information	Optional
Service request	Direct	putCISServiceRequest	Push delivery of a service request, which has to be delivered from service requester to the service provider. CIS information is delivered in the specific container to address the instruction to process information and/ or implement the requested service. Exchange information such as requester and service provider identification and exchange status are provided to easy controls.	Message container including: payload (optional when not delivered in former data delivery exchange) + CIS information (mandatory service request information) + exchange information with relevant information to enable exchange features	N
Service feedback	Return	putCISServicefeedback	Push delivery of a CIS service feedback. CIS service feedback information is delivered in the specific container to address the status and result of processing and /or and/or implementation of the requested service. Exchange information such as client and supplier identification and exchange status may be provided to easy exchange related controls.	Message container including: payload (optional when not delivered in data delivery exchange) + CIS information (mandatory service feedback information) + exchange information with relevant information to enable exchange features	N
Exchange information return	Return (direct on service feedback)	D2Exchange Infomation	Exchange information is returned from client to supplier to provide return status i.e. success, fail and exchange context information to easy controls such as supplier and client identification.	Exchange information	N

10.3 State diagrams

State diagrams are not needed and not developed for stateless FEP+EP as simple CIS.

10.4 Features implementation description

10.4.1 Overview

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the simple push data exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

10.4.2 Subscription contract

10.4.2.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in a client to assess the identity of supplier and authenticate the supplier request in messages exchange.

10.4.2.2 Catalogue

Managed offline, not automated.

10.4.3 Session

10.4.3.1 Session life cycle

No session is managed for the current EP+FEP.

10.4.3.2 Link monitoring

Not managed in this EP+FEP.

10.4.4 Information management

10.4.4.1 Overview of information management features

CIS features implementation is based on service request and service feedback implementation.

As explained in Annex G, besides the exchange actors' involvement, collaborative ITS services workflow management to support information process and transactions features involves application level management. These application level checking and triggers, which are needed to implement control and triggers to start exchange features, are considered to be existing and are not described in this document, which only assumes these processing and triggering functions are needed and implemented at service requester system and service provider system and are only indicated in the sequence diagrams as placeholders to support understanding of exchange workflows.

NOTE 1 Application level triggers and checks depend on the specific payload domain which is managed in the exchange and can be specified only based on the payload content itself, which is out of scope of this document. Examples of domain-specific information are referred to in the basic exchange data model CIS information (see Annex C) as predefined services such as: VMS message processing, information broadcast delivery or traffic management plans activation and implementation.

Figure 32 describes the interaction for implementing a service request from the service requester system to one up to many service provider systems by the simple CIS exchange.

After the application level service requester system has triggered the service request exchange management, at first outcome it will receive the information that a service request has been delivered to the involved service provider systems, or some errors have occured or have been recognized by the service provider systems in the service request itself. Returned exchange information is the first outcome of the service request delivery to the service provider and service provider system.

For CIS, it is also assumed that at the service requester system some application level management of service requests which have been delivered is needed and feedback and error/time-out management is done as application level implementation. The outcomes of such management will lead an application logic to decide on the next workflow and exchange action needed, but this is not described in this document.

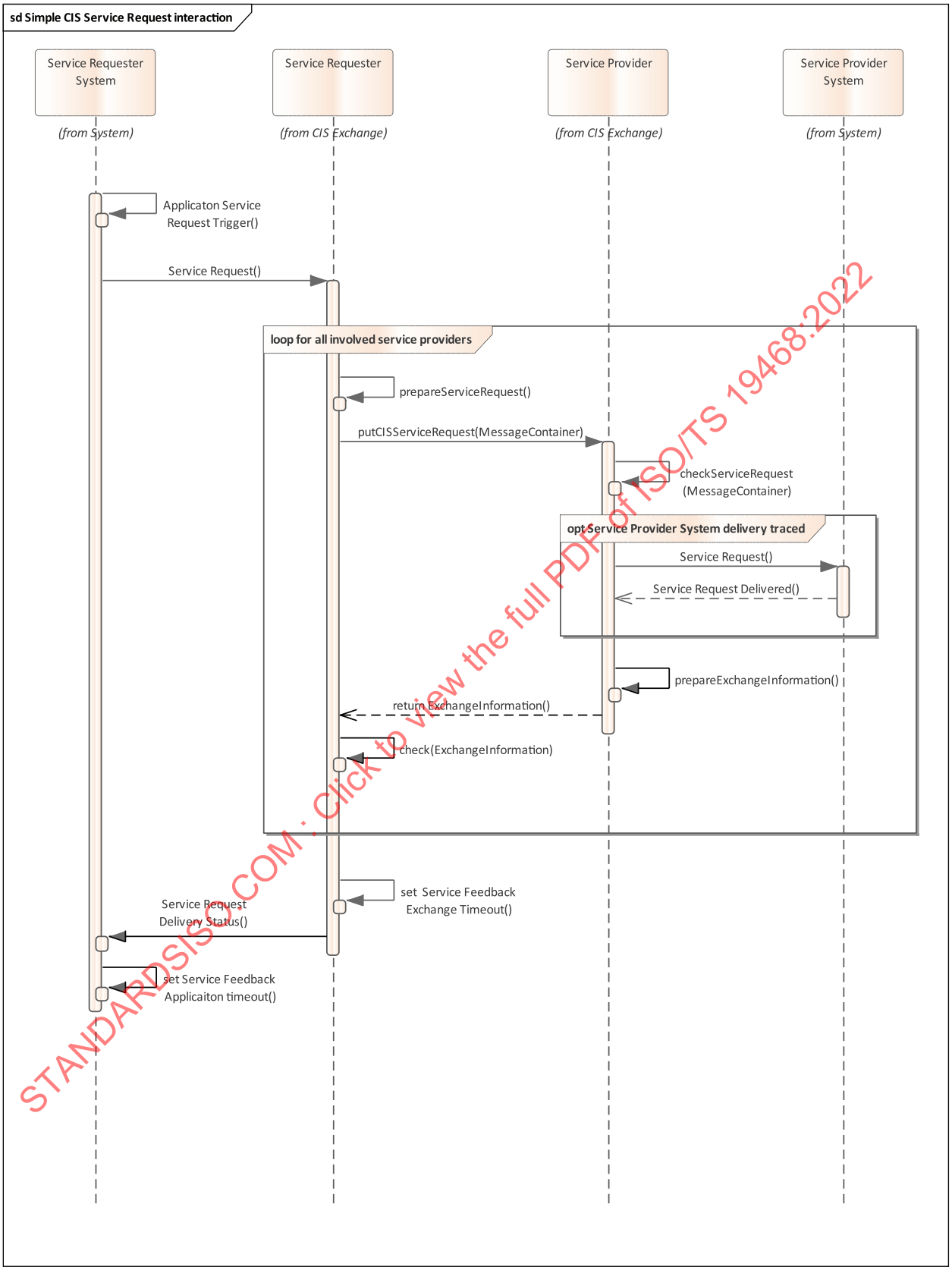

Figure 32 — Simple CIS sequence diagram for service request

As stated above, only interaction among exchange interfaces, i.e. service requester and service provider are considered normative for the specific FEP+EP in this document. Other interactions are assumed but

are dependent on application logic and requirements which are external to the exchange system; these details are not within the scope of this document. General aspect for CIS and some application level workflow requirements are described in Annex G.

Figure 33 describes the interaction for implementing a service feedback from the service provider system to the service requester system by the simple CIS exchange. After the application level service provider system has delivered the feedback to the service requester exchange management, at first outcome it will receive the information that service feedback has been delivered to the service requester system or that errors have occured or have been recognized by the service requester systems in the service feedback itself. Returned exchange information is the first outcome of the service feedback delivery to the service requester and service requester system.

For CIS, it is also assumed that at service requester system some application level management of service feedbacks which have been delivered is needed and feedback and error/time-out management is done as application level implementation. The outcomes of such management will lead an application logic to decide on the next workflow and exchange action needed, but this is not described in this document.

NOTE 2 Service requests and service feedbacks are asynchronous and multiple service feedbacks are possible after one service request. Figure 33 shows frame "alt involved service provider gives feedback" as a single step of processing of the service request which can be repeated since the processing has come to an end. The processing can also be influenced by subsequent service requests in the processing phase such as a processing update or process cancellation request.

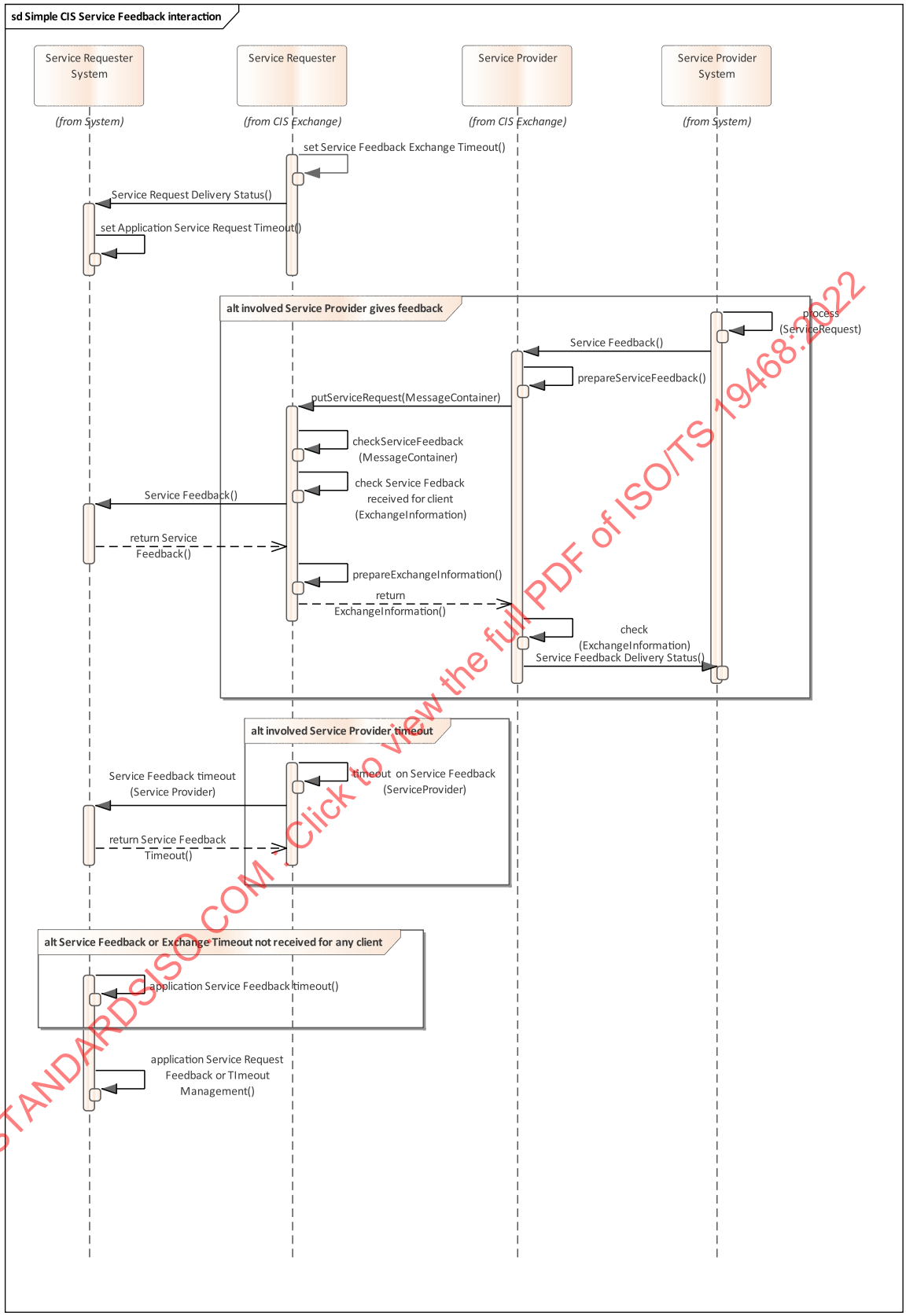

Figure 33 — Simple CIS sequence diagram for service feedback

10.4.4.2 Support information processing

While the workflow to enable the support to information processing is defined in 10.4.4.1, the information needed to enable such information processing is described in the basic exchange data model in Annex C, specifically under the CIS information description.

Predefined services are possible as well as not-predefined services, which are enabled by specific attributes in the CIS information classes.

10.4.4.3 Distributed transaction

While the workflow to enable the support to distribute not-atomic transactions processing is defined in 10.4.4.1, the information needed to enable such information processing is described in the basic exchange data model at Annex C, specifically under the CIS information description.

Predefined services are possible as well as not-predefined services, which are enabled by specific attributes in the CIS information classes.

10.4.4.4 Data delivery

Not managed in this EP+FEP.

In simple CIS FEP+EP PIM, it is assumed that data delivery requirements to deliver data to be processed are enabled by a parallel data delivery exchange and referenced in CIS information to specify the relative processing action. Despite that payload data can also be delivered in a service request or service feedback when needed by specific application level management, it can be assumed as being delivered in a previous exchange. The service requester system itself is assumed to be in charge of responsibility of data availability to support actions and processing requests.

10.4.5 Self-description

Handshake not available.

10.4.6 Communication/protocol

Communication features are implemented at PSM level. They are relevant to the specific platform chosen on which the exchange pattern will be implemented (e.g. http/XML, Web Services SOAP, REST).

11 Stateful CIS

11.1 Overview

Stateful CIS is an extension of the simple CIS, described and specified for the simple CIS FEP+EP PIM (i.e. Clause 10 in this document), with session management and link monitoring features which are enabled by the same mechanisms which are defined for the stateful push FEP + EP PIM (i.e. Clause 9 in this document). The overall implementation of stateful CIS will be a combination of features described for both simple CIS service request and service feedback mechanism, with session management and link monitoring features implemented as per stateful push.

Only a specific stateful push definition will be described in this clause. For all definitions and specifications which are the same as for stateful push or simple CIS FEP+EP PIM, the user is addressed to the specific sections in this document.

NOTE All definitions related to simple CIS and stateful push FEP + EP PIM apply, but supplier role and name are associated to the requester system and client role and name are associated to the provider systems.

Stateful CIS FEP+EP PIM is based on the description of interactions which enable service request and feedback exchange among a service requester and one to many service providers as illustrated in Annex G.

It can be implemented in several technological platforms with specific interactions methods, e.g. SOAP, WebService methods.

The stateful CIS FEP+EP frameworks enables a common communication interface to embed ITS service request and feedback in a way that enables two or more TMC, TIC or SP systems to perform in a common, interoperable way.

Stateful CIS FEP+EP framework is not designed to implement data delivery business scenarios but it may be based on payload content which is assumed to be exchanged by Data Delivery which can be enabled by Data Delivery FEP+EP. Only Specific features enabling CIS are described and introduced for Simple CIS. Feature which are not related to this FEP+EP are martked as "Not applicable" in Table 9.

To describe the stateful CIS EP+FEP at PIM level all features are described in a general abstract format, independently from the specific technology platform in which this model will be implemented. (e.g. http/get XML, WebService). See Table 15.

Table 15 — Selection of features for Stateful CIS

Features area	Feature	Simple push available
Subscription contract	Contract	N
	Catalogue	N
Session	Session life cycle	Y
	Link monitoring	Y
Information management	Operating modes	Not applicable
	Update methods	Not applicable
	Life cycle management	Not applicable
	Support information processing	Y
	Distributed transaction	Y, not atomic transactions
Data delivery	Data delivery	Not applicable
	Data request	Not applicable
	Large datasets handling	Not applicable
	Synchronization	Not applicable
Self-description	Handshake	N
Communication	Security	At PSM level
	Compression	At PSM level
	Communication	At PSM level

11.2 Exchange pattern and messages definition

11.2.1 Overall presentation

A simple CIS stateful CIS FEP+EP enables one service requester node to implement CIS service request interaction for one up to several service providers. At the same time, session management and link monitoring features allows stateful CIS service requester and service provider systems to be aware of network communication features among them to enable reliable and timely exchange and take initiative to trigger any required actions besides CIS actions themselves in case a session or link are broken by system or network failures.

The interaction is described as the CIS service requester addressing all involved CIS service providers through their stateful CIS exchange interfaces which provides methods for delivering service requests from a requester to a multiplicity of service providers (Figure 34).

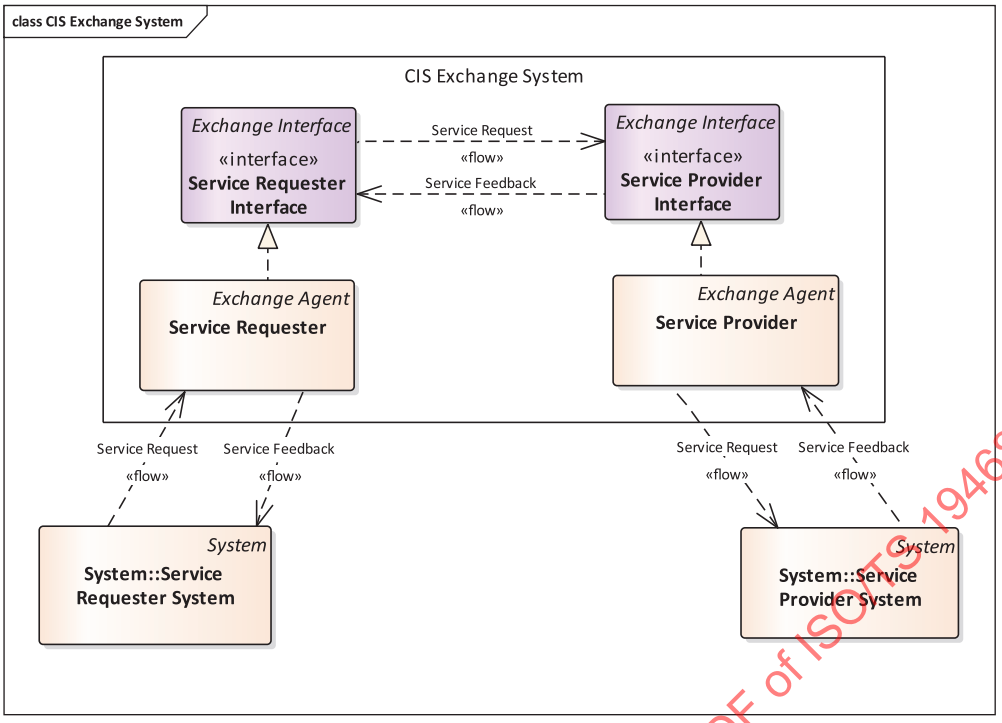

Figure 34 — Stateful CIS exchange actors

The simple CIS exchange pattern is described in the following subclauses.

11.2.2 Basic exchange pattern

In a stateful CIS context, the service provider provides a mechanism to receive a service request from action taken at a service requester site. This will result in the service requester invoking specific methods /resources offered by the service provider. On the reverse side, a mechanism is also needed to enable the service provider to feed back the result of processing or action triggered by the service request, so the service requester itself provides a mechanism to receive a service feedback from the involved service providers when their actions or processes results are available.

Besides simple CIS service requests and service feedback methods, the stateful CIS client also provides a keepAlive method to implement link monitoring capabilities among service requester and service providers. The keepAlive method is used from the service requester to advise the client when no service request are to be delivered, so the service requester delivers a keepAlive message to check and enable the service providers to verify that exchange systems and network connection are available, despite the service requester not needing to exchange payload content. KeepAlive messages are delivered by the service requester to the service providers, according a time interval which is defined among them.

Stateful CIS session management methods are available to implement session management features in the same way they are described and available for stateful push FEP+EP PIM; such methods inlcude namely openSession and closeSession usage, which are used to define dynamic exchange information context to enable session management exchange features.

Figure 35 shows the communication diagram for stateful CIS FEP+EP.

In this FEP+EP framework, the service requester delivers a service request trigger to the service provider and the service provider delivers service feedback information to the service requester.

Therefore, the stateful CIS service requester logically pushes CIS service requests messages, embedded in a MessageContainer, to the stateful CIS service provider, which shall acknowledge the reception of such messages by a return message, embedded in an ExchangeInformation. This exchange information return message is available to bring information back from the service provider to the service

requester, such as failure or success. Return message information is logically described in this PIM, while implementation will be defined at PSM level.

At the same time, the involved stateful CIS service providers logically push CIS service feedback messages, embed in a message container to the stateful CIS service requester which symmetrically shall acknowledge the reception of such messages by a return message, embedded in an ExchangeInformation. This exchange information return message is available to bring some information back to the service provider, the management of which is not relevant to this FEP+EP and is not described in this document.

The management of any further workflow based on any processing or action errors is in this workflow framework pattern based on CIS service feedback. It is only in charge of the service requester system and is defined at application level based on the specific application requirements needed to enable the specific collaborative ITS services.

In the context of this stateful push FEP+EP framework, to enable interoperability among the CIS service requester and CIS service providers, all rules defined in this subclause apply.

Any simple CIS service provider exchange system shall realize a stateful CIS service provider interface which provides a putCISServiceRequest method, and an openSession, closeSession and keepAlive methods.

Any simple push service requester exchange system shall realize a simple CIS service requester interface which can invoke the putCISServiceRequest, openSession, closeSession and keepAlive methods provided by the simple CIS service provider interface to deliver CIS service requests to trigger an action/process by the service provider systems.

Any simple CIS service requester exchange system shall realize a simple CIS service requester interface which provides a putCISServiceFeedback method.

Any simple CIS service provider exchange system shall realize a simple CIS service provider interface which can invoke the putServiceRequest method provided by the simple CIS service requester interface to deliver CIS service feedback to the service requester systems.

Figure 35 shows the communication diagram for simple push FEP+EP.

In this FEP+EP framework, the service requester pushes service requests messages to the service provider and the service provider pushes service feedback messages to the service requester.

Both service provider and service requester shall acknowledge the received service request or service feedback messages by a return information to the corresponding counterpart, i.e. respectively to the service requester and to the service provider. This return information shall be coded as ExchangeInformation.

NOTE This return message is available to bring some exchange information back from the receiver system which can be used for any further exchange features implementations or application level checks or processing and/or workflow management decision in CIS implementation which are out of scope of this document.

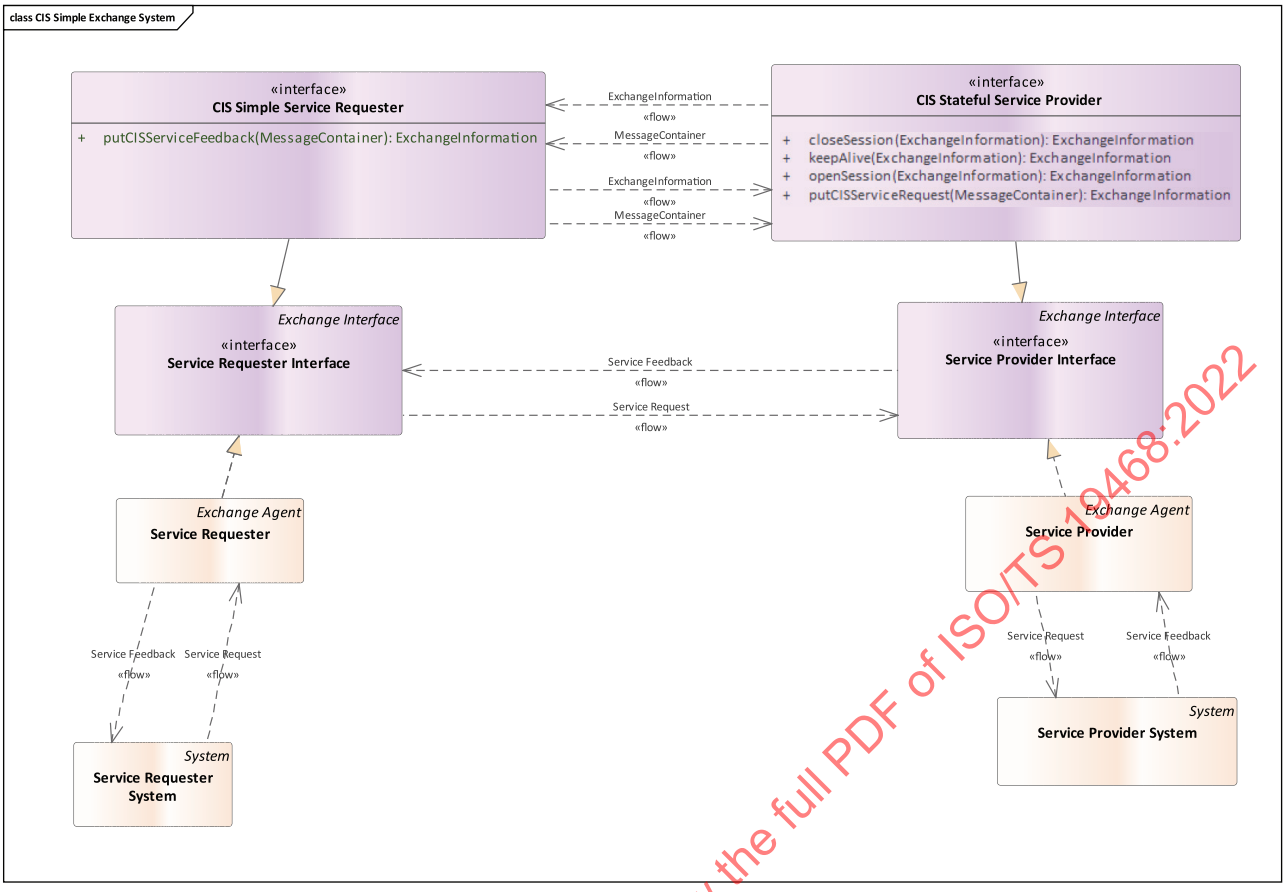

Figure 35 — Stateful CIS exchange subsystems, interface interactions and methods

11.2.3 Relevant exchange information from exchange data model

11.2.3.1 Overview

A basic exchange data model has been provided to allow the implementation to enable CIS features to support information processing and enable distributed transactions features.

CIS information embeds all information needed to request for specific processing and to enable workflow management to support transaction negotiation processes by service request and service feedback messages.

Exchange information enables delivery of the exchange context of the involved service requester and service providers to enable orchestration of ITS services supplying and exchanging dynamic information which enables general exchange status information to monitor transactions negotiation or service processing in general.

11.2.3.2 Exchange information

Some non-mandatory information which should be managed in the exchange information for easy application development is fully described in the basic exchange data model:

— Service provider identification

  — Requirement: supplier identification. (String)

— Client identification

  — Requirement: client identification. (String)

— Exchange information (provided both by the client and the supplier) wraps exchange and exchange status information "Success", "Fail", "Close Session Request", “SessionId”.

— Generation timestamp information

  — Requirement: timely, reliable information.

11.2.3.3 CIS information

— Service request

  This provides the reference to elements to be processed and the requested action, also addressing the service for which the action is triggered.

— Predefined service

  — Requirements: support for information processing, distributed transaction.

— Requested action

  — Requirements: support for information processing, distributed transaction, distributed atomic transaction.

Service feedback

  — Requirements: support for feedback on information processing, distributed transaction.

Service request status

  — Requirements: support information processing, support for feedback on information processing, distributed transaction.

11.2.4 Exchanged messages

Different messages or supplier/client interactions (invoked method) are exchanged in simple push which are needed to manage synchronization, payload exchange and link monitoring. These are formally contained in pushed messages to a client from a supplier or in return messages from a client to a supplier. See Table 16.

Table 16 — List of messages types and detailed content

Interaction message	Direction requester provider	Designation	Description	Exchanged information	Optional
Open session	Direct	openSession	Service requester initializes a stateful CIS session with the involved service provider.	Exchange information	N

Table 16 *(continued)*

Interaction message	Direction requester provider	Designation	Description	Exchanged information	Optional
Service request	Direct	putCISServiceRequest	Push delivery of a service request, which has to be delivered from the service requester to the service provider. CIS information is delivered in the specific container to address the instruction to process information and/or implement the requested service. Exchange information such as requester and service provider identification and exchange status are provided to easy controls.	Message container including: payload (optional when not delivered in former data delivery exchange) + CIS information (mandatory service request information) + exchange information with relevant information to enable exchange features	N
Service feedback	Return	putCISServicefeedback	Push delivery of a CIS service feedback. CIS service feedback information is delivered in the specific container to address the status and result of processing and / or implementation of the requested service. Exchange information such as client and supplier identification and exchange status may be provided to easy exchange related controls.	Message container including: payload (optional when not delivered in data delivery exchange) + CIS information (mandatory service feedback information) + exchange information with relevant information to enable exchange features	N
KeepAlive	Direct	keepAlive	Test exchange link and confirm session validity when no service request or feedback is exchanged. It shall deliver the session ID of a previously opened session, wrapped in exchange information.	Exchange information	N
Close session	Direct	closeSession	Message to gracefully close a stateful CIS session, initiated by the service requester.	Exchange information	N

Table 16 *(continued)*

Interaction message	Direction requester provider	Designation	Description	Exchanged information	Optional
Exchange information return	Return (Direct on service feedback)	D2Exchange Infomation	Exchange information is returned from client to supplier to provide return status, i.e. Success, Fail, and exchange context information to easy controls such as supplier and client identification.	Exchange information	N

11.3 State diagrams

The service requester initiates the communication and by open session and keepAlive messages can be aware of the service provider status based on return messages or possible communication errors.

Figure 36 describes the internal service requester status and the corresponding service provider status as monitored and managed by the service requester.

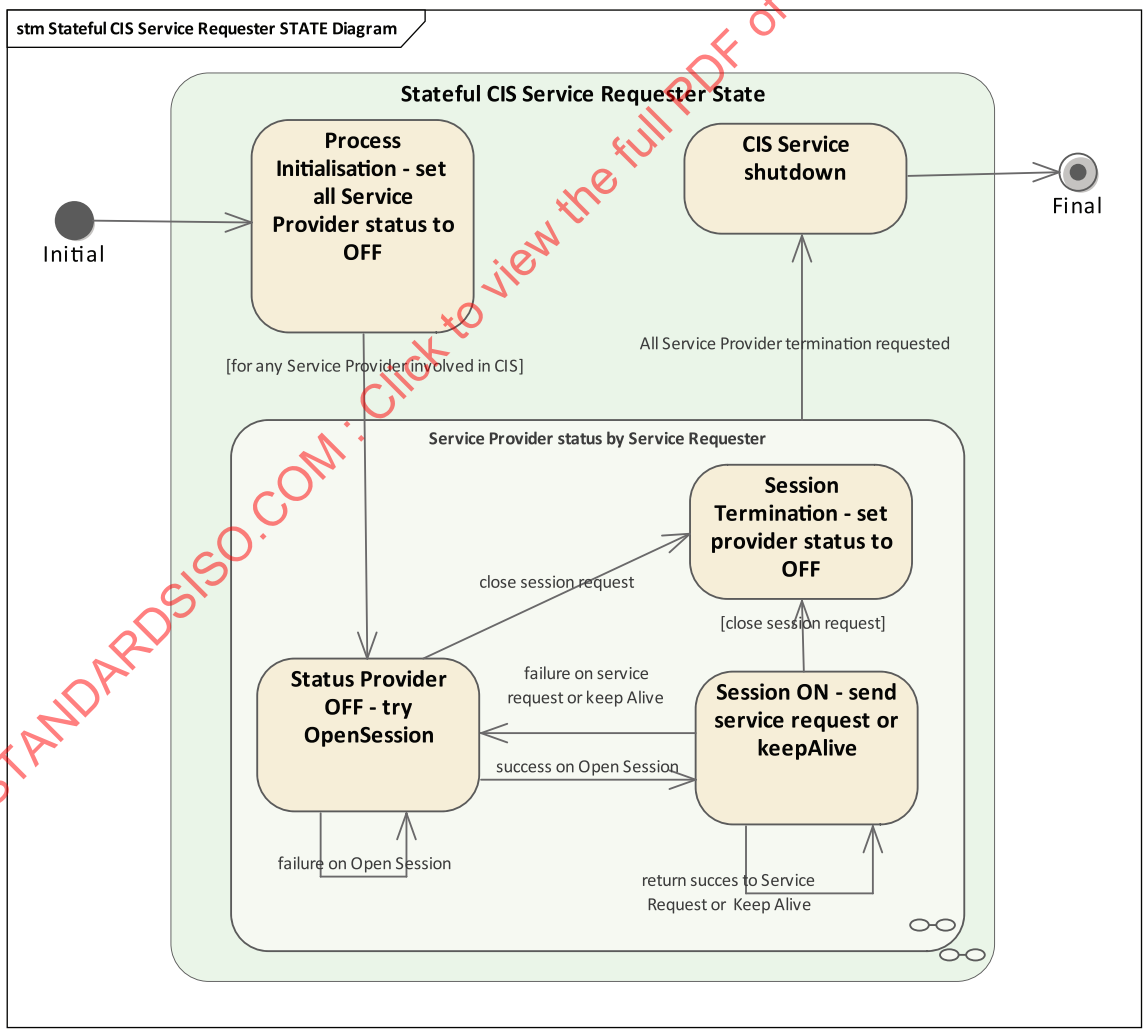

Figure 36 — CIS requester-side Stateful CIS state diagram

Figure 37 describes the internal service provider status and the service requester status as monitored and managed by the service provider.

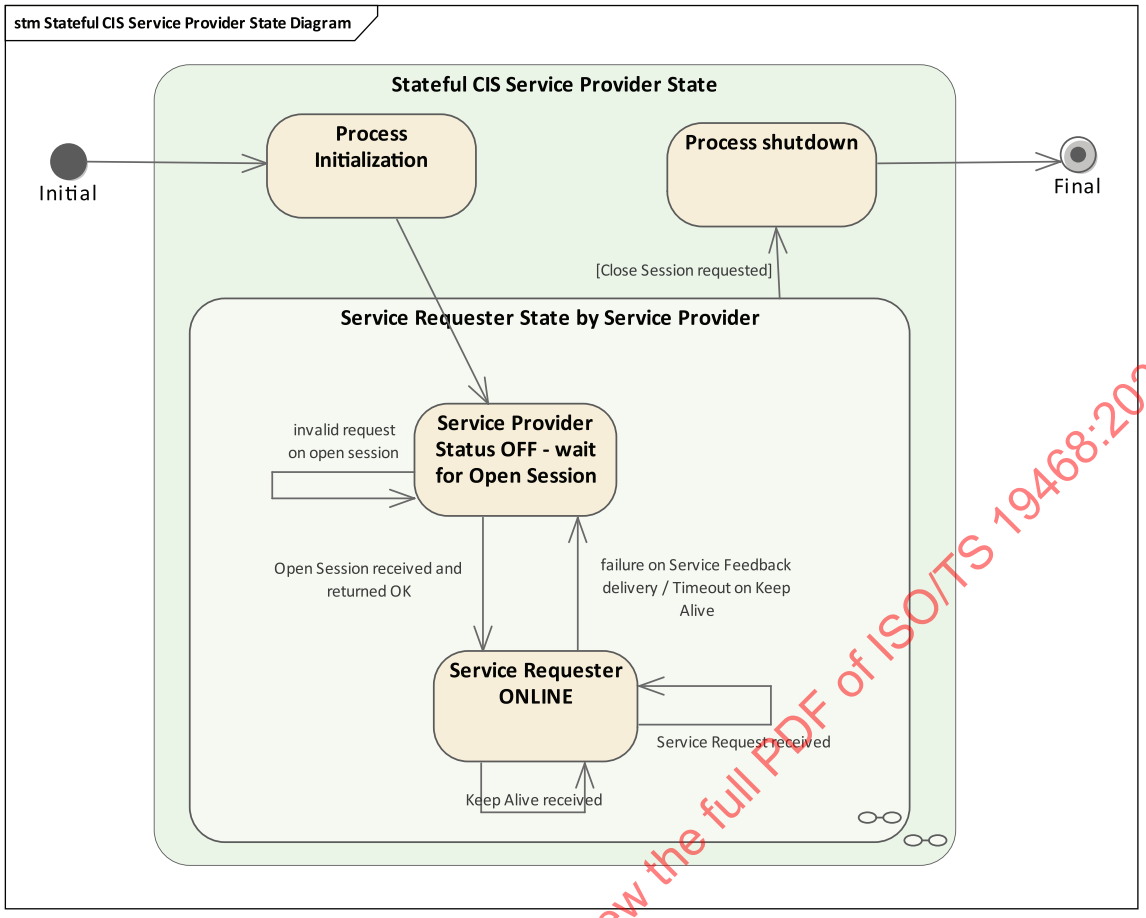

Figure 37 — CIS Provider-side Stateful CIS state diagram

Specific management in the initialization and termination of a push process should be considered at the application level in the supplier and client systems.

11.4 Features implementation description

11.4.1 Overview

This subclause provides a description and the corresponding specification for each feature identified in the context diagram, according to the simple push data exchange architecture. The following features are specified:

— subscription contract;

— subscription (also known as session);

— information management;

— data delivery;

— communication/protocol.

11.4.2 Subscription contract

11.4.2.1 Contract

Managed offline, not automated. It assumes information for controls to be implemented in a client to assess the identity of supplier and authenticate the supplier request in messages exchange.

11.4.2.2 Catalogue

Managed offline, not automated.

11.4.3 Session

11.4.3.1 Session life cycle

After the session state management diagrams, at exchange initialization a session needs to be initiated. An open session is tried in loop until it succeeds.

A failure when opening a session includes cases of a client who has not subscribed or is not authorized. Checking this can be ensured at the PSM level (e.g. this could include VPN setting or IP firewall or signatures handling). Logical information may be included in exchange data to be managed in subscription check at the service provider side.

When a session is "on" the service requester may send a service request to the involved service providers when all conditions to start a CIS service are triggered.

In case no CIS service requester triggering condition occur, a keepAlive message is used to check session status for the supplier and the client.

When no keepAlive message or no payload is received, after a keepAlive time-out, the service provider invalidates the session on its side and returns a "close session" message at first received message of any type.

If any service request or keepAlive fails, the service requester invalidates the session and starts a new loop to open a session.

Any message and return in the sequence diagram (Figure 37) will be mapped in PSM definition to a real platform available implementation such as a web service "service request and return" or any other available mechanism in a specific platform.

11.4.3.2 Link monitoring

After the keepAlive mechanism has been introduced in the previous session life cycle, service requests and keepAlive messages are exchanged which give evidence to the service requester and service providers about the session and systems status.

When no service request or feedback are exchanged, and a time-out has expired, a keepAlive message is exchanged to check the network and system availability.

In case service request or service feedback or keepAlive fail, the session shall be invalidated.

11.4.4 Information management

11.4.4.1 Overview of information management features

Besides the service request and service feedback handshaking, which is fully described in the equivalent stateful CIS subclause, stateful CIS implements these concepts in the framework of a stateful session. This implementation notifies the application-side service requester about the availability and session status of all service providers which are involved in the collaborative ITS service implementation.

This enables the decision process to trigger and manage CIS to involve information about involved CIS service provider status and availability.

All application level rules about decisions to trigger services and involved providers' orchestration functionalities of CIS are out of scope of the exchange specification and are not described in this document.

11.4.4.2 Support information processing

See 10.4.4.2.

11.4.4.3 Distributed transaction

See 10.4.4.3.

11.4.4.4 Data delivery

Not managed in this EP+FEP.

In stateful CIS FEP+EP PIM it is assumed that data delivery requirements to deliver data to be processed are enabled by a parallel data delivery exchange and referenced in CIS information to specify the relative processing action. Despite that payload data can also be delivered in a service request or service feedback when needed by specific application level management, it can be assumed as being delivered in a previous exchange. The service requester system itself is assumed to be in charge of responsibility of data availability to support actions and processing requests.

11.4.5 Self-description

Handshake not available.

11.4.6 Communication

Communication features are implemented at PSM level. They are relevant to the specific platform chosen on which the EP will be implemented (e.g. http/XML, Web Services SOAP, REST).

12 Other PIM definitions

This clause is reserved for future new models/patterns supporting existing or new exchange business scenarios.

The definition of the business scenario is mandatory prior to the PIM description because a PIM implementation is connected to a business scenario.

Annex A
(informative)

Methodology presentation

A.1 Introduction

This annex presents the approach followed for this document and the rationale behind the choices made.

A.2 Apply model-driven architecture

The original approach taken for modeling the full set of aspects covered by data exchange clearly identified the need to separate the exchange specifications, of what needs to be exchanged and the exchange of the data itself. This is reflected by creating two independent interoperability domains:

— The information interoperability domain as reflected in the PIM defining content (e.g. of DATEX II);

— The exchange interoperability domain as reflected by this document.

While the payload content model can be regarded as describing "what to exchange", the present exchange specification deals with the problems about "how to exchange".

A distinction has also been made in the modeling phase to separate the abstract model from its concrete implementation(s). In practice, this approach led to the creation of a PIM for exchange, which described the concepts behind exchange, and one or more PSMs which defined how the abstract model would actually be implemented on specific technical platforms. As this basic principle of the Model Driven Architecture (MDA) has guided all the work that was already undertaken for creating the full set of specifications for content definition with notable success, it was chosen as well for the Exchange Specification definition.

Therefore, this exchange specification is based on a PIM for exchange (the Exchange PIM), which is detailed in one or more documents containing specifications targeted to specific platform implementations (PSM).

A.3 Use case-driven

One of the principles followed in this document is to clearly identify the business scenarios that are addressed by the specifications plus the full set of features that can be available for implementing actual systems based on this document for each of those business scenarios. This led to the identification of two main business scenarios:

— Information delivery;

— Collaborative ITS services.

The information delivery business scenario addresses the exchange of traffic and travel information between two data exchange nodes, whereas the collaborative ITS services business scenario broadens this approach by including the possibility of having one data exchange node stimulating actions within another data exchange node by requesting the execution of a particular service offered by this node.

These two business scenarios clearly show distinct characteristics, but in order to fully describe them, both need to be detailed further, leading to different possible options. Therefore, the approach was to

further refine these general business scenarios into particular, more detailed use cases, each of which address specific requirements.

The term use case is used here to describe a set of interactions between entities (called actors) and the system being analysed, providing a better understanding of the main functions behind such interactions.

In the case of the information delivery domain, the actors involved in the corresponding use cases are the supplier of exchanged information and the receiver of such information – the client. For the collaborative ITS services domain, the actors involved are the entity requesting the ITS services, or the service consumer, and the entity providing the service, i.e. the service provider.

A.4 Functional exchange profiles

When analysing the business scenarios and requirements for this document, it became apparent that different use cases within such a business scenario might contain disparate requirements, which could not be easily accommodated into a single, specific implementation. Even worse, some use cases might even contain requirements that are, by nature, contradictory, depending on the business needs of the user community they originated from.

With that in mind, this specification identifies the full set of features that can be available for a data exchange process to take place. Then, in a second step, the specification reflects the particular use cases and selects the best suited set of features for each of them. This selection is denoted a functional exchange profile (FEP).

A.5 Profile-to-platform mapping

By performing a short survey on the capabilities that modern ICT platforms offer, it became apparent that some of the features indicated for implementing data exchange systems can be realized differently depending on the platforms chosen. As such, the one-size-fits-all approach that was followed on the previous version of the Exchange Specification for creating concrete implementations simply does not fit into the new paradigm.

At the heart of the current specification sits the PIM for data exchange. Its purpose is to model the interactions that were identified for each use case in an abstract, platform-independent way. The exchange PIM is detailed in this document.

This platform-independent specification then has to be mapped to a realization of the indicated FEP on a specific technical platform described in a PSM for data exchange (ISO 14823 defines such a mapping, for example). The act of matching an FEP to a platform is also known as profile-to-platform mapping. Each one of these mappings form what is called an interoperability domain. It is only in the scope of such an interoperability domain that two data exchange nodes can expect to be interoperable, i.e. if two different implementers of data exchange systems need to be sure that their system will be able to communicate, they both need to follow the same mapping.

Annex B
(normative)

Definition of requirements

B.1 Information requirements

The primary intent of the information delivery scenario is the provision of information by a supplier to a client. Table B.1 provides requirements that are related to the data actually being transferred from supplier to client. For each requirement it is stated whether it is applicable to data delivery or collaborative ITS services (CIS) business scenarios, assuming that the CIS supplier is intended as service requester and client as service provider and data are exchanged or processed as input to provision of services.

Table B.1 — Information requirements

Requirement	Description	Data delivery	Collaborative ITS services
Simple server	In a simple exchange, the supplier keeps no track of previous interactions that have occurred between any of its clients. Therefore, the client is responsible for providing the supplier with all information it can need in order to serve its request. Note that this requirement does not oblige the supplier to have a mechanism for receiving state information as that depends on the availability of other requirements described in this clause (e.g. filter handling).	Y	
Stateful server	Any supplier implementing this feature should provide some sort of state-keeping mechanism that will allow it to know what information was sent to its client. Therefore, it would be possible for the supplier to send only information that has not been already sent before. A supplier implementing this mechanism can also support other requirements that allow clients to shape the actual data set being received (e.g. filter handling).	Y	Y
Subscription	The information that a supplier delivers to a client is defined by a subscription. A subscription results from an interchange/licence agreement, where both parties agree on parameters like, for example, the information type and periodicity that should be observed upon data delivered. The supplier can choose to reject requests without proper subscription, or it can deliver a standard set of information.	Y	Y
Catalogue exchange	The information a supplier provides is described in a catalogue. The catalogue is used to identify the information contained in a subscription.	Y	Y
Self-description	The ability of two nodes to exchange information that will allow them to negotiate the requirements supported by both of them (handshake). This feature deals with exchange parameters, which is different from a catalogue exchange that deals with content.	Y	Y

Table B.1 *(continued)*

Requirement	Description	Data delivery	Collaborative ITS services
Filter handling	A supplier implementing this feature enables clients to provide specific filters that shall be applied to the information being exchanged.	Y	Y
Client profiles	A client profile lets suppliers shape the information they send according to the requirements of the client requesting it.	Y	Y
Interchange/Licence agreement	Legal artefact where parties taking part in data exchange define the terms and conditions that govern the whole exchange process (e.g. Non-disclosure of information, service level agreements, etc.).	Y	Y
Integrity	This feature implies that the data that is prepared by the supplier should reach its intended recipient without being tampered with in any way, semantic, structural or other.	Y	Y
Full audit trail data delivery (all state changes)	All data item versions are delivered to the client.	Y	Y
Snapshot data delivery (last known state)	Only the current version of the data items is delivered to the client.	Y	Y
Incremental data delivery	A mechanism where the server sends only the information that has changed since the previous exchange cycle	Y	Y
Reference datasets for different versions	Service a supplier should provide for the client to access referenced data in a versioned way. Even if new versions appear, the old versions remain accessible.	Y	Y
Extensibility	Extensions to the data exchange protocol should be supported; therefore, any implementation of the data delivery use case shall take into account that the protocol can evolve. Thus, each message should state the protocol version it refers to.	Y	Y
Support for life cycle management	A set of functions for updating information at client systems according to updated information gathered at supplier system.	Y	Y
Support for information processing	A set of functions that will let the supplier (as service requester) tell the client (as service provider) what to do with the information being exchanged (processing directive).		Y
Support for feedback on information processing	A set of functions that will let the client (service provider) tell the supplier (service requester) the outcomes of processing the information by the stated directive which had been exchanged.		Y
Distributed transaction	A capability for the exchange system to coordinate among several involved service provider systems to collaborate in implementing a distributed service.		Y
Distributed atomic transaction	A capability for the exchange system to coordinate among several involved service provider systems to collaborate in implementing a distributed service keeping consistency in transaction.		Y
Synchronization	A mechanism that lets clients request the whole set of information currently known to the supplier to ensure that its internal data structures be in exactly the same state as those of the supplier (intended for CIS, the service provider needs the exact information to provide the requested services).	Y	Y

Table B.1 *(continued)*

Requirement	Description	Data delivery	Collaborative ITS services
Delayed delivery	In the case that preparing and sending a data set by the supplier would take too much time to complete, the supplier should inform the client about this fact. This mechanism should also define how and when the client would be able to access the information it needs.	Y	
Data delivered as soon as possible	This feature is used to ensure that the supplier sends the information as soon as it becomes available in its system.	Y	Y
On demand request (query)	The ability of the client to ask the server for information it needs whenever it wants (Client pull).	Y	

B.2 Communication requirements

Communication requirements characterize the mechanism that data exchange nodes implement in order to address problems specific to the communication layer of the data exchange process. These requirements are completely unaware of both the security and information features that are in use. The idea is that a given supplier has prepared a particular dataset; the requirements described in Table B.2 can be used to successfully convey it to the client.

Table B.2 — Communication requirements

Requirement	Description	Data delivery	Collaborative ITS services
Sessionless	No session is used during the data exchange process.	Y	Y
Session	Client and supplier negotiate and establish a session before starting to exchange information. The session parameters constitute state information shared between both stations.	Y	Y
Request/response	A mechanism where the client requests the data and instantly receives the response.	Y	Y
Delivery/response	A mechanism where the supplier initiates the data delivery process, while the client patiently waits for it.	Y	Y
Error handling	A mechanism that allows both client and supplier to detect that an error has occurred during the exchange process and to decide what actions should be taken to handle it.	Y	Y
Timely responses	The communication layer should introduce a minimum delay on the data delivery process, ideally none.	Y	Y
Time-out management	The ability to handle time-out situations that happen during an exchange process.	Y	Y
Exchange quality measures (e.g. response timestamp)	The messages exchanged should include extra information that allows both parties involved to measure the quality of service of the communication layer.	Y	Y
Logging	A mechanism for storing information about exchange activities, which could then be used to analyse the whole process.	Y	Y
Failed data recovery	When the supplier fails to deliver the information to the client, this feature ensures that the failed data messages will be successfully delivered to the client at a later time.	Y	Y

Table B.2 *(continued)*

Requirement	Description	Data delivery	Collaborative ITS services
Message sequence	A mechanism that allows identifying each message exchanged between two entities by including a unique sequence number.	Y	Y
Full reliability	A mechanism that ensures that data sent by a supplier is really received by the client provided that both client and supplier are active and have the ability to communicate. This does not include any semantic validation. Syntax validation is optional.	Y	Y
Link monitoring and control	This feature enables both parties to continually check whether the communication link works properly and act accordingly when it is broken.	Y	Y
On occurrence update	A supplier implementing this feature should send the information to the client as soon as it is available.	Y	
Periodic update	A supplier implementing this feature should buffer all the information to be sent to a particular client for a pre-defined time period and send information only when this period has elapsed.	Y	
Multi-part data delivery	When the size of the information to be delivered is too large, the supplier can choose to deliver it in chunks. At the first request, the supplier returns the first data chunk. The response contains the message ID and the total number of parts (chunks) that comprise the dataset. The client then has to request each of the remaining parts of the message.	Y	Y
Compression	The ability to pack the same information in a smaller amount of data in order to decrease the transmission time.	Y	Y

B.3 Security requirements

The requirements described in Table B.3 deal with all aspects used to provide security services at any of the different communication levels, such as peer authentication, channel security, etc.

Table B.3 — Security requirements

Requirement	Description	Data delivery	Collaborative ITS services
Client authentication	The act of establishing or confirming a client as authentic.	Y	Y
Client authorization	The process of verifying if a client is allowed to access a resource (commonly referred to as read access) or execute an action (commonly referred to as write access).	Y	Y
Supplier authentication	The act of establishing or confirming a supplier as authentic.	Y	Y
Supplier authorization	The process of verifying if a supplier is allowed to access a resource (commonly referred to as read access) or execute an action (commonly referred to as write access).	Y	Y
State of the intended recipient	The act of indicating the destination peer of a message.	Y	Y

Table B.3 *(continued)*

Requirement	Description	Data delivery	Collaborative ITS services
Confidentiality	Ensuring that information is accessible only to those authorized to have access (ISO/IEC 27002).	Y	Y
Client identification	A mechanism that allows a client to provide their identity.	Y	Y
Supplier identification	A mechanism that allows a supplier to provide their identity.	Y	Y
Non-repudiation	A mechanism to guarantee that the sender (supplier of a client) of a message cannot later deny having sent the message.	Y	Y

B.4 Financial/economic requirements

Although not at the same level, the requirements described in Table B.4 have some economic/financial impact on the actual implementation of the data exchange sub-system.

Table B.4 — Financial/economic requirements

Requirement	Description	Data delivery	Collaborative ITS services
Reasonable TCO (total cost of ownership)	The data exchange sub-system should have a reasonable TCO (total cost of ownership).	Y	Y
Expandability at a reasonable cost (scalability)	The data exchange sub-system should be implemented in such a way that it can be possible to increase the capacity of the system at a reasonable cost. Capacity relates to any of the system resources, such as data volumes, computation power, parallel processing, etc.	Y	Y
Low processing resources	It shall be possible to implement a data exchange sub-system on systems with low processing resources.	Y	Y

Annex C
(normative)

Basic exchange data model and data dictionary

C.1 Overall presentation

The data model described is based on a UML methodology that is independent from any technical platform.

Whenever exchange features as session management, link monitoring is necessary and some extra information needs to be conveyed among supplier and client to enable control and reliable exchange management. The basic exchange data model describes the more common data needed to implement the exchange features supported by the different exchange patterns.

As in specific contexts more than one information payload needs to be exchanged, this model further allows the exchange of multiple payloads based on such extra exchange requirements.

This model is suggested for the implementation of minumum exchange information and to enable interoperability among exchange interfaces which implement the same exchange pattern, but can be unnecessary for specific simplified PSMs implicitely assuming the few mandatory data defined in the model (e.g. supplier identification, protocol type, protocol version, exchange status).

C.2 Basic exchange data model

C.2.1 Overview

The basic exchange data model is used to convey information which is needed to manage exchange features such as information management, session management, link monitoring.

Further information is required to manage CIS.

C.2.2 The MessageContainer class diagram

A MessageContainer class (see Figure C.1) is introduced to allow delivery of further information besides payload such as:

— Exchange information, which is needed to manage exchange features such as session management and link monitoring and error management.

— Information management-related information.

— Collaborative ITS services information.

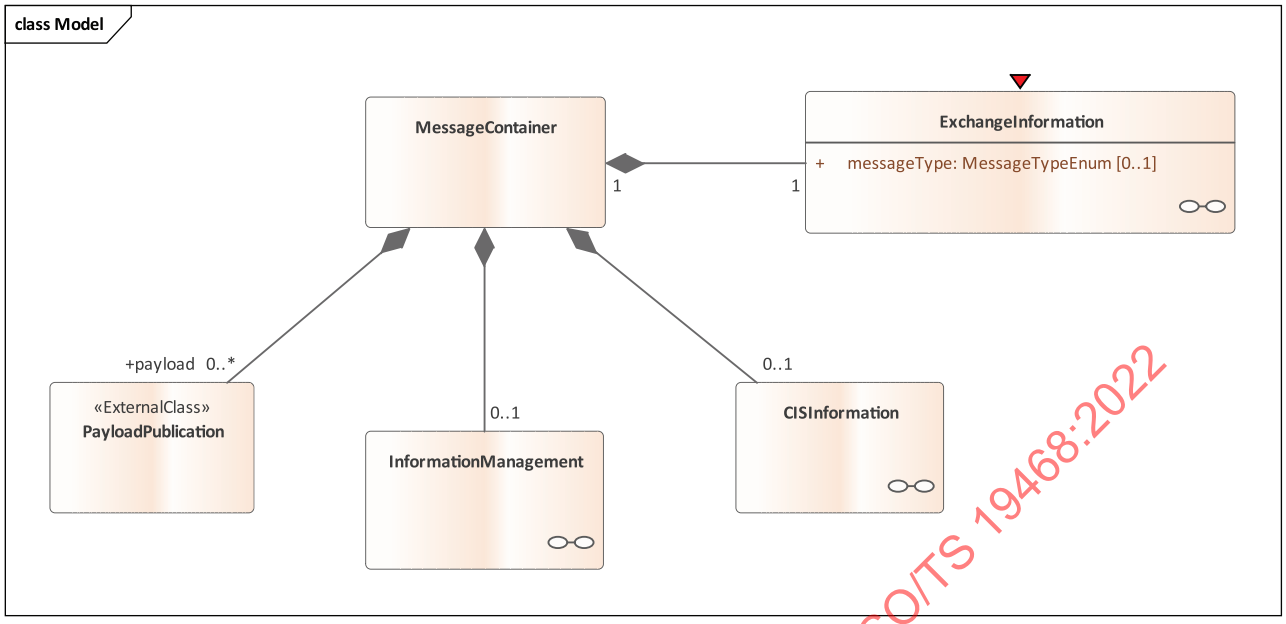

Figure C.1 — The MessageContainer class diagram

Classes linked to information management and CIS information classes are optional, while minimum exchange information is to be managed, for example, to convey information about the used protocol and version and supplier identification as exchange responsibility. Other constraints on implementation depending on different FEP+EP which implement the exchange and its features, for example exchange dynamic information attributes, will be mandatory in a FEP+EP which wants to implement session management, such as stateful push FEP+EP.

The MessageContainer class also allows conveying multiple payloads within a single exchanged message.

C.2.3 The ExchangeInformation class diagram

The exchange information class (Figure C.2) is used to convey information about the exchange environment such as:

— Exchange context: this is implied by supplier and client(s) identification, represented as exchange agents, which may be associated to an external international identifier, the type of protocol that is defined to be used in exchanging data, the implemented version of this protocol, the operating mode and updated method used in the data delivery exchange.

— Subscription, with its period validity information and delivery frequence. An external enhanced validity may support more information, such as recurring periodic validity for specific time intervals within specific days.

— Dynamic information, such as exchange status and return status (within a set of predefined exchanged status and return status enumeration lists and their reason) and SessionID for exchange patterns which manage sessions. Sequencing number for synchronization purposes and a Boolean information about payload completion enable large dataset handling features available in some exchange patterns.

— Return information, used in return messages to convey a first information about information delivery and CIS request exchange and also to send back special requests to manage such as snapshot synchronization.

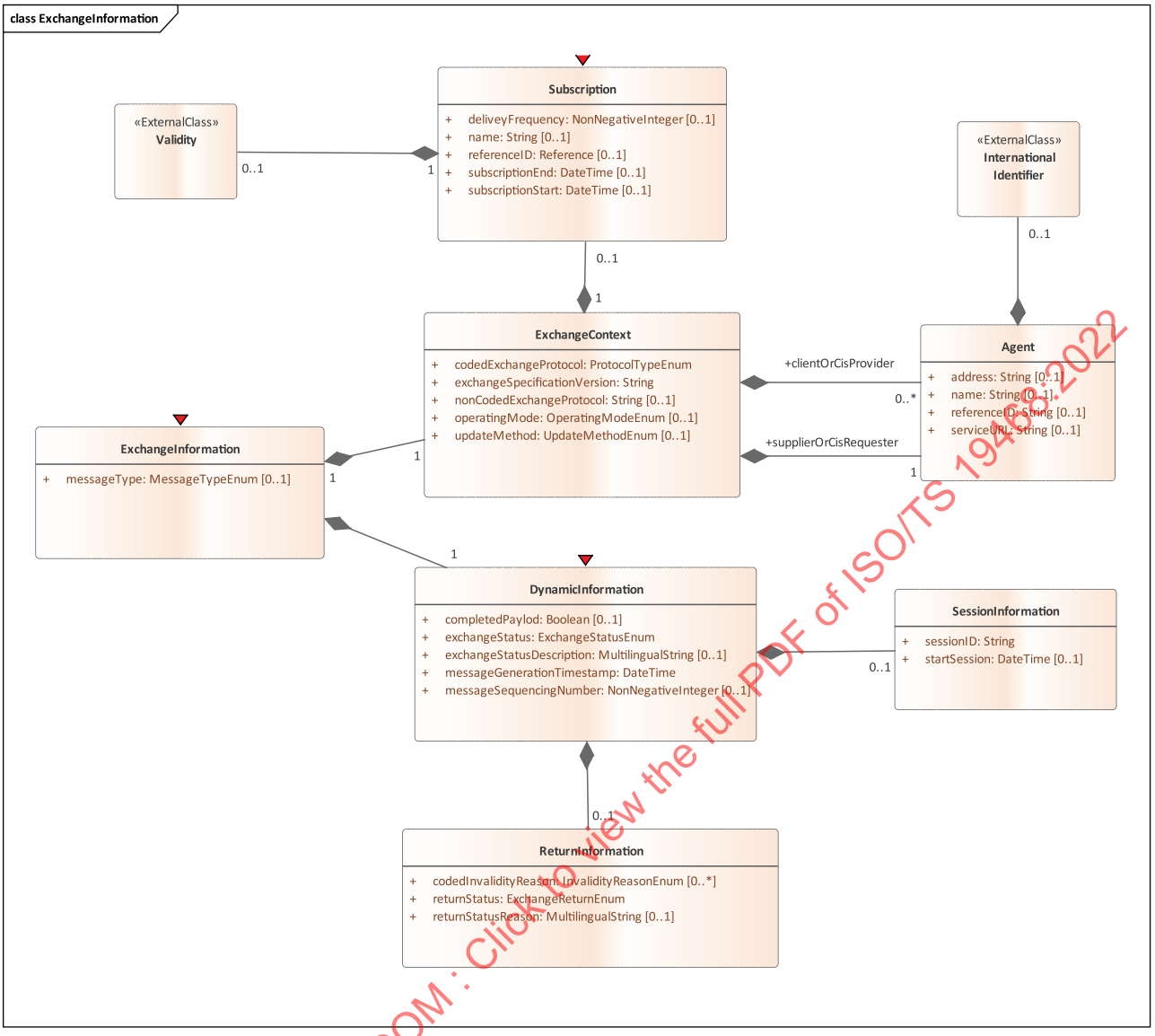

Figure C.2 — ExchangeInformation class diagram

Exchange context is mostly implicit in the subscription contract, but it is instantiated to simplify or enable the application level check for some communication features such as supplier and client identification, authentication and authorization, which can only be implemented in a reliable way based on communication layers.

In dynamic information, the attribute exchangeStatus is the only mandatory information, but as this status may be not managed, its value can be set to "undefined". Other information will be de facto mandatory whenever needed to be managed in specific FEP+EP, for example. in case of session management feature implementation, SessionID attribute in class SessionInformation will be needed to be managed.

C.2.4 The InformationManagement class diagram

The InformationManagement class (Figure C.3) allows delivering information management data: as defined in Annex F for life cycle information, only valid information is managed in the payload. Whenever an element ends its life cycle, i.e. being closed or cancelled, it is no longer conveyed in the payload and information management is to be delivered to manage closures or cancellation of such elements.

These closed and cancelled elements are conveyed linked through InformationManagedResourceList class and are addressed by ElementReference class, which allows identification of the element by its reference or versioned reference, specifying its managementStatus, i.e. closed or cancelled.

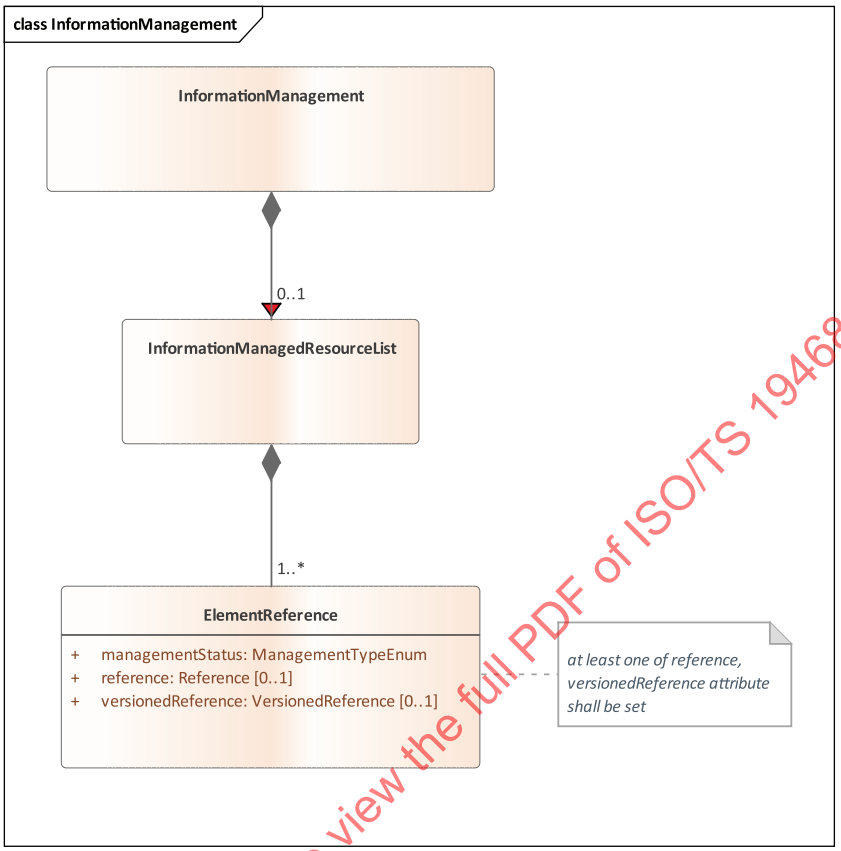

Figure C.3 — InformationManagement class diagram

C.2.5 The CISInformation class diagram

With reference to Annex G, collaborative ITS services transform the vision of supplier and client in service provider and service requester: the CISInformation class (Figure C.4) conveys in the data model the information needed to implement service request and service feedback as described to be exchanged between service provider and service requester.

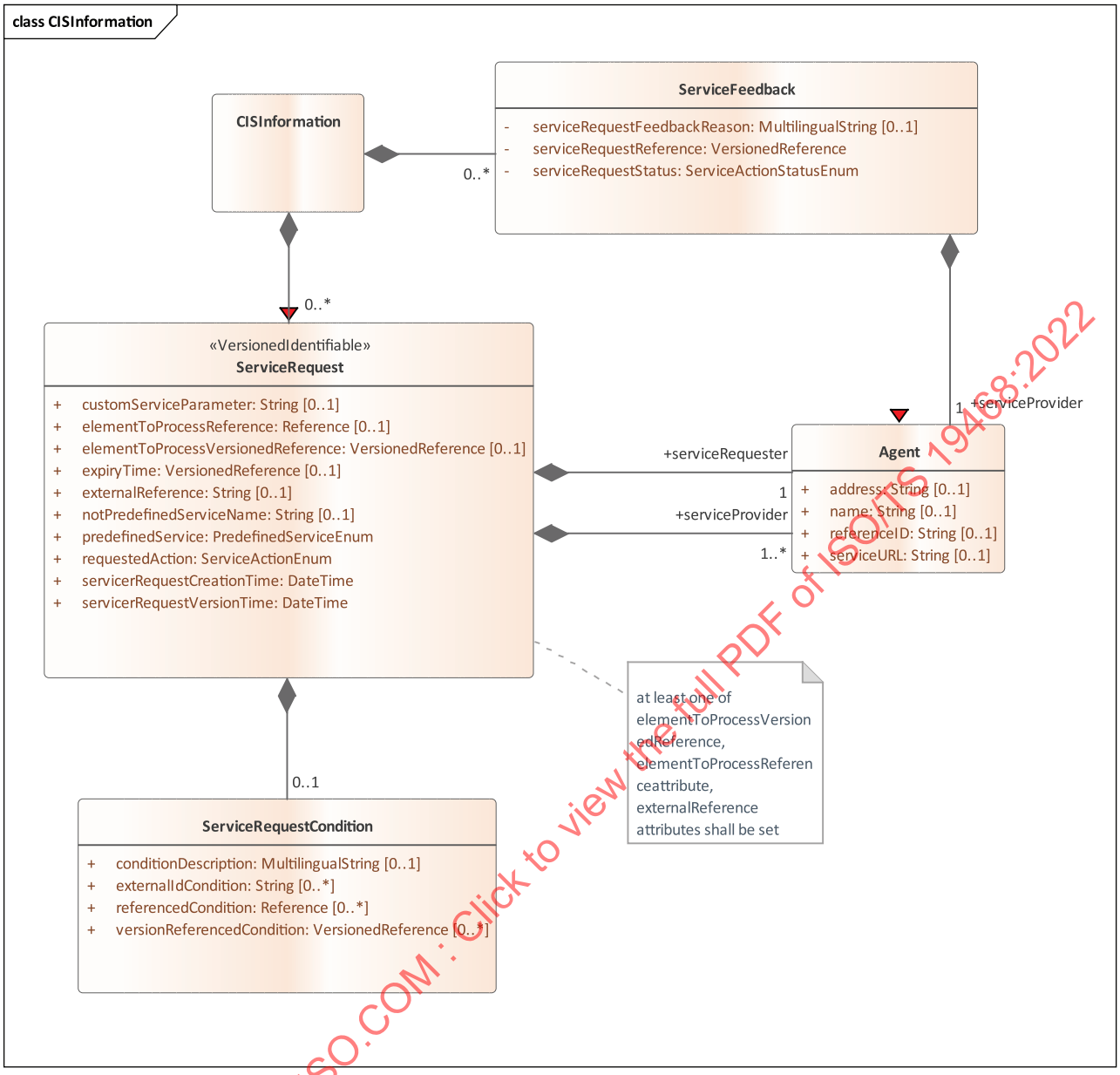

Figure C.4 — The CISInformation class diagram

CISInformation class: allows any Service Request and Service Feedback to be conveyed which are requested to be exchanged among Service Requester and Service Providers through the respective named classes ServiceRequest and ServiceFeedback.

ServiceRequest: conveys information related to a single service request:

— service request creation and update time,

— predefinedService: to be chosen among an enumerated list such as processing, broadcast delivery, VMS message process, TMP activation,

— not-predefined service name,

— custom service parameter: custom parameter to raise for processing,

— requestedAction: approve, implement, terminate, cancel,

— element to process reference or external reference,

— expiry time when a service is valid only implemented within a limited time amount after which it will be no longer needed.

Service request can be related to a service condition through the ServiceCondition class identifying the needs for that kind of service through some payload reference for a predefined, coded or not-predefined (i.e. unpredictable) condition.

ServiceRequest class also allows addressing the identification of involved service providers needed to implement such requests and the corresponding service.

ServiceFeedback: allows information to be conveyed about the processing of a service request previously exchanged through reference information to the service request itself and specifying its status among a set (compliant, failed, not compliant, processing, rejected, scheduled, success, timedOut) plus a reason for that status.

C.3 Data dictionary overview

This data dictionary describes the characteristics of the different classes, attributes and roles appearing in the data model defined in C.2. The dictionary is specified as a set of tables grouping classes, attributes and roles for each package as they are defined in C.2.

For each package the following are successively provided:

a) the class table,

b) the association table,

c) the attribute table.

Each table of a given type has the same structure.

The data dictionary is categorized into sections following the different UML model packages as mentioned above. It defines for every package the entities and elements corresponding.

The table columns have the following meaning:

a) Name: it provides the symbolic name (either in lower or upper camel case) given to the corresponding class, attribute or association role.

b) Designation: it provides the corresponding name in natural language of the corresponding class, attribute or role.

c) Definition: it provides a comprehensive definition detailing this class, attribute or association role.

Some columns are specific for one or two tables. The class tables include the following column:

d) Abstract: it indicates if the corresponding class is abstract (value “yes”) or concrete (value “no”). Abstract classes are defined in ISO/IEC 19505-1.

The attribute tables and the association tables include the following column:

e) Multiplicity: it provides the number of occurrences a class may have when instantiating this association (resp. a class attribute may have when instantiating this class). The adopted syntax is the following: m..n where ‘m’ and ‘n’ respectively represent the minimum and the maximum value of multiplicity.

   For association roles, the possible values for ‘m’ are:

   1) 0 in case of an optional participation of the corresponding class when instantiating the association;

2) 1 in case of a mandatory participation of the corresponding class when instantiating the association;

3) 2, 3, ... in case a minimum number of participations of the corresponding class is explicitly defined when instantiating the association.

For association ends, the possible value for ‘n’ are:

4) 1 in case only one class instance is at most participating at the association instantiation;

5) * in case several instances are allowed to participate at the association instantiation;

6) 2, 3, ... in case a maximum number of participations of the corresponding class is explicitly defined when instantiating the association.

For attributes, the possible values for 'm' are:

7) 0 in case of an optional attribute;

8) 1 in case of a mandatory association/attribute;

9) 2, 3, ... in case a minimum number of occurrences is explicitly defined.

For attributes, the possible value for 'n' are:

10) 1 in case only one attribute instance is allowed;

11) * in case several instances are allowed for this attribute without being specified;

12) 2, 3, .. in case a maximum number of occurrences is explicitly defined.

For the attribute tables, the following column has been added:

f) Type: it provides the class name used as data type. It is only provided for elements corresponding to class attributes. When the type name ends with 'Enum' this means it corresponds to an enumeration of accepted values defined in C.6.

For the association table, the following column has been added:

g) Target: it provides the class name appearing at the second end of the relationship, i.e. linked through the corresponding association.

C.4 Data dictionary for "ExchangeDataModel"

C.4.1 InformationManagement "Classes" package

C.4.1.1 Location of "Classes" package

The location of "Classes" package is:

— MessageContainer/InformationManagement/Classes

C.4.1.2 Classes of the "Classes" package

Table C.1 shows classes of the "Classes" package.

Table C.1 — Classes of the "Classes" package

Class name	Designation	Definition	Abstract
ElementReference	Element reference	Element reference	no
InformationManagedResourceList	Information managed resource list	Managed resource list	no
InformationManagement	Information management	Information management	no

C.4.1.3 Associations of the "Classes" package

Table C.2 shows associations of the "Classes" package.

Table C.2 — Associations of the "Classes" package

Class name	Association end	Designation	Definition	Multiplicity	Target
InformationManagedResourceList	elementReference	Element reference		1..*	ElementReference
InformationManagement	informationManagedResourceList	Information managed resource list		0..1	InformationManagedResourceList

C.4.1.4 Attributes of the "Classes" package

Table C.3 shows attributes of the "Classes" package.

Table C.3 — Attributes of the "Classes" package

Class name	Attribute name	Designation	Definition	Multiplicity	Type
ElementReference	managementStatus	Management status	It identifies the status of the element referenced	1..1	ManagementTypeEnum
	reference	Reference	It identifies an element reference	0..1	Reference
	versionedReference	Versioned reference	It identifies an element versioned reference	0..1	VersionedReference

C.4.2 ExchangeInformation "Classes" package

C.4.2.1 Location of "Classes" package

The location of "Classes" package is:

— MessageContainer/ExchangeInformation/Classes

C.4.2.2 Classes of the "Classes" package

Table C.4 shows classes of the "Classes" package.

Table C.4 — Classes of the "Classes" package

Class name	Designation	Definition	Abstract
Agent	Agent	an actor in the exchange context	no
DynamicInformation	Dynamic information	dynamic exchange information	no
ExchangeContext	Exchange context	exchange context, e.g. which defines the specific EP and FEP and other details about supplier and client	no
ExchangeInformation	Exchange information	exchange information	no
ReturnInformation	Return information	the information provided as return after a message has been delivered	no
SessionInformation	Session information	session information	no
Subscription	Subscription	a subscription between a supplier and its client or among service providers and service requester in the collaborative ITS services framework	no

C.4.2.3 Associations of the "Classes" package

Table C.5 shows associations of the "Classes" package.

Table C.5 — Associations of the "Classes" package

Class name	Association end	Designation	Definition	Multiplicity	Target
Agent	internationalIdentifier	International identifier		0..1	InternationalIdentifier
DynamicInformation	returnInformation	Return information		0..1	ReturnInformation
	sessionInformation	Session information		0..1	SessionInformation
ExchangeContext	subscription	Subscription		0..1	Subscription
	clientOrCisProvider	Client or CIS provider	it defines the client or CIS provider information of the exchange context; depending on EP it can be instantiated for single or multiple client or no one	0..*	Agent
	supplierOrCisRequester	Supplier or CIS requester	it defines the supplier or CIS requester information of the exchange context	1..1	Agent
ExchangeInformation	dynamicInformation	Dynamic information		1..1	DynamicInformation
	exchangeContext	Exchange context		1..1	ExchangeContext
Subscription	validity	Validity		0..1	Validity

C.4.2.4 Attributes of the "Classes" package

Table C.6 shows attributes of the "Classes" package.

Table C.6 — Attributes of the "Classes" package

Class name	Attribute name	Designation	Definition	Multiplicity	Type
Agent	address	Address	the network address of the exchange agent	0..1	String
	name	Name	name of the agent in the exchange context	0..1	String
	referenceID	Reference id	a reference for the agent in the exchange context	0..1	String
	serviceURL	Service url	the URL to address the agent service	0..1	String
DynamicInformation	completedPaylod	Completed paylod	attribute which can be used to indicate when a payload has been completed or not within the current message; when set to false, the following messages will deliver and complete all the payload content relative to the current exchange or current session	0..1	Boolean
	exchangeStatus	Exchange status	status of exchange defined by protocol specification	1..1	ExchangeStatusEnum
	exchangeStatusDescription	Exchange status description	multilingual textual status description of exchange defined by protocol specification	0..1	MultilingualString
	messageGenerationTimestamp	Message generation timestamp	the date time in which the message had been generated	1..1	DateTime
	messageSequencingNumber	Message sequencing number	a number, always increasing within a same session among a client and supplier, which can be used to order message within a delivery	0..1	NonNegativeInteger
ExchangeContext	codedExchangeProtocol	Coded exchange protocol	the exchange protocol type as referenced by any standard or by agreement among client and supplier, e.g. snapshot pull, simple push, collaborative ITS services, etc.	1..1	ProtocolTypeEnum
	exchangeSpecificationVersion	Exchange specification version	the version of the protocol used for the exchange according to a standard or as agreed among client and supplier	1..1	String
	nonCodedExchangeProtocol	Non coded exchange protocol	when a protocol is used in the exchange which is not predefined coded protocol, this attribute defines protocol information among supplier and client	0..1	String
	operatingMode	Operating mode	feature which specifies when the information should be delivered	0..1	OperatingModeEnum

Table C.6 *(continued)*

Class name	Attribute name	Designation	Definition	Multiplicity	Type
	updateMethod	Update method	exchange feature used in the protocol which specifies the rules to generate the information payload exchanged	0..1	UpdateMethodEnum
ExchangeInformation	messageType	Message type	the message type which is used in the specific exchange pattern to define the use of exchanged message, e,g, payload delivery, open session, keepAlive, CIS service request and feedback etc.	0..1	MessageTypeEnum
ReturnInformation	codedInvalidityReason	Coded invalidity reason	specifies the invalid information which has been found in a message by the receiver	0..*	InvalidityReasonEnum
	returnStatus	Return status	the return status of a message previously delivered	1..1	ExchangeReturnEnum
	returnStatusReason	Return status reason	the reason for the setting of the return status	0..1	MultilingualString
SessionInformation	sessionID	Session id	the ID of session established among client and supplier	1..1	String
	startSession	Start session	the start date and time of the session	0..1	DateTime
Subscription	deliveyFrequency	Delivey frequency	the planned time payload delivery frequence as number in seconds; it includes keepAlive messages delivery when no payload is to be delivered	0..1	NonNegativeInteger
	name	Name	the descriptive name of the subscription	0..1	String
	referenceID	Reference i d	a reference identification for the subscription for the exchange	0..1	Reference
	subscriptionEnd	Subscription end	defines the date and time when the subscription is to be considered ended	0..1	DateTime
	subscriptionStart	Subscription start	defines the date and time when the subscription is to be considered active	0..1	DateTime

C.4.3 CISInformation "Classes" package

C.4.3.1 Location of "Classes" package

The location of "Classes" package is:

— MessageContainer/CISInformation/Classes

C.4.3.2 Classes of the "Classes" package

Table C.7 shows classes of the "Classes" package.

Table C.7 — Classes of the "Classes" package

Class name	Designation	Definition	Abstract
CISInformation	CIS information	CIS information	no
ServiceFeedback	Service feedback	feedback about a specific service request from the service implementer to the requester	no
ServiceRequest	Service request	service request from the service implementer to the requester	no
ServiceRequestCondition	Service request condition	specifies the condition which is behind the need for the service request, e.g. a specific situation or situation record, travel times status, specific road data or external conditions	no

C.4.3.3 Associations of the "Classes" package

Table C.8 shows associations of the "Classes" package.

Table C.8 — Associations of the "Classes" package

Class name	Association end	Designation	Definition	Multiplicity	Target
CISInformation	serviceFeedback	Service feedback		0..*	ServiceFeedback
	serviceRequest	Service request		0..*	ServiceRequest
ServiceFeedback	ServiceProvider	Service provider	identifies the list of the service provider agents of the service feedback	1..1	Agent
ServiceRequest	serviceRequestCondition	Service request condition		0..1	ServiceRequest-Condition
	serviceRequester	Service requester	identifies the service requester agent of the service request	1..1	Agent
	ServiceProvider	Service implementer	identifies the list of international identifier implementer of the service request	1..*	Agent

C.4.3.4 Attributes of the "Classes" package

Table C.9 shows attributes of the "Classes" package.

Table C.9 — Attributes of the "Classes" package

Class name	Attribute name	Designation	Definition	Multiplicity	Type
ServiceFeedback	serviceRequestFeedbackReason	Service request feedback reason	additional text to feedback about the status of the service request	0..1	MultilingualString
	serviceRequestReference	Service request reference	reference to the service request to which the service feedback refers	1..1	VersionedReference
	serviceRequestStatus	Service request status	specifies the Status of Service request referenced	1..1	ServiceActionStatusEnum
ServiceRequest	customServiceParameter	Custom service parameter	a string conveying information for custom parameter to service	0..1	String
	elementToProcessReference	Element to process reference	element reference to be processed	0..1	Reference
	elementToProcessVersionedReference	Element to process versioned reference	element versioned reference to be processed	0..1	VersionedReference
	expiryTime	Expiry time	date time until which the required action for service is to be implemented	0..1	VersionedReference
	externalReference	External reference	external reference to be processed	0..1	String
	notPredefinedServiceName	Not predefined service name	name of service not predefined	0..1	String
	predefinedService	Predefined service	type of predefined service	1..1	PredefinedServiceEnum
	requestedAction	Requested action	identifies the action requested for the specified service	1..1	ServiceActionEnum
	servicerRequestCreationTime	Servicer request creation time	time of creation request	1..1	DateTime
	servicerRequestVersionTime	Servicer request version time	time of version request time	1..1	DateTime
ServiceRequestCondition	conditionDescription	Condition description	a multilingual description of the condition under which the service request is instantiated	0..1	MultilingualString
	externalIdCondition	External id condition	en external reference ID to the condition for the service request	0..*	String
	referencedCondition	Referenced condition	the list of condition information which is referenced by an identifiable in payload publications	0..*	Reference

Table C.9 *(continued)*

Class name	Attribute name	Designation	Definition	Multiplicity	Type
	versionReferencedCondition	Version referenced condition	the list of condition information which is version referenced by an identifiable in payload publications	0..*	VersionedReference

C.4.4 "Common" package

C.4.4.1 Location of "Common" package

The location of "Common" package is:

— MessageContainer/Common

C.4.4.2 Classes of the "Common" package

Table C.10 shows classes of the "Common" package.

NOTE This table is intentionally empty. It is a void container used to import classes from external projects; no classes are defined in this document but a common class hook is kept.

Table C.10 — Classes of the "Common" package

Class name	Designation	Definition	Stereotype	Abstract

C.4.4.3 External classes of the "Common" package

These classes are not part of the exchange packages as such but are generally provided through the payload content.

Table C.11 shows external classes of the "Common" package.

Table C.11 — External classes of the "Common" package

Class name	Designation	Definition
InternationalIdentifier	International identifier	an identifier/name whose range is specific to the particular country
PayloadPublication	Payload publication	a payload publication of domain specific information created at a specific point in time that can be exchanged at an exchange interface
Validity	Validity	specification of validity

C.4.4.4 Associations of the "Common" package

There are no defined associations in the "Common" package.

C.4.4.5 Attributes of the "Common" package

There are no defined attributes in the "Common" package.

C.4.5 "MessageContainer" package

C.4.5.1 Location of "MessageContainer" package

The location of "MessageContainer" package is:

— MessageContainer

C.4.5.2 Classes of the "MessageContainer" package

Table C.12 shows classes of the "MessageContainer" package.