The Product Catalog as Foundation for Modern Airline Retailing

A Shared IT Providers Perspective on Capabilities, Dependencies, and Transition Considerations

IATA, June 2026

Abstract

The transformation toward Modern Airline Retailing with Offers & Orders introduces new requirements for how Products are defined, managed, and exposed across the value chain. Across the industry, fragmented Product terminology, tightly coupled legacy constructs, and inconsistent interfaces continue to limit agility, slow innovation, and increase integration complexity for airlines and their partners. At the centre of this transformation lies the Product Catalog, which is widely recognised as a foundational capability enabling Product consistency, interoperability, and scalability across commercial, operational, and distribution contexts.

This document presents a consolidated perspective from IT Providers participating in an IATA-led industry sprint initiative, reflecting observed approaches to Product Catalog implementations. It identifies common capabilities, recurring building blocks, and key dependencies, and analyses interactions with adjacent domains such as Offer Management, Order Management, Stock management, and downstream fulfilment systems. The document is intended as a non-prescriptive industry reference rather than a target architecture, standard, or implementation guideline.

Recognising that airlines and ecosystem partners operate in heterogeneous technology environments, the document does not propose a single target solution. Instead, it highlights areas of convergence, captures variation across implementations, and identifies the conditions required to enable interoperability, including stable Product References (ID + version), consistent Product terminology, and a shared minimum data layer. It also outlines key transition considerations, including the treatment of legacy constructs as output-only, transitional elements and the need for controlled coexistence between legacy and modern systems.

Through this consolidated view, the document aims to support a shared industry understanding, enable airlines and value-chain stakeholders to assess current capabilities and dependencies, and identify areas where further alignment or convergence may be required to sustainably support Modern Airline Retailing with Offers & Orders at scale.

Table of Contents

Abstract 2

Table of Contents 3

1. Introduction 4

2. Product Catalog-related Terminology in Use 5

3. Product Catalog Data Element Concepts and Observed Relationships 7

4. Product Catalog Interactions and Observed Responsibilities 9

5. Conceptual Models of Product Catalog Implementations 12

6. Common Building Blocks of Product Catalog Implementations 16

7. Key Dependencies Between Product Catalog Building Blocks 18

8. Interfaces and Information Exchange Between Product Catalog and Other Systems 20
   - 8.1 Minimum Product Information Across Interfaces 21
   - 8.2 Reference-Based Access and Interaction Patterns 21
   - 8.3 Vendor-Specific and Context-Dependent Elements 22
   - 8.4 Level of Detail for Interoperable Exchange 22

9. Transition Considerations for Product Catalog Implementations 23
   - 9.1 Legacy Constructs (RFISC, SSR, Fare Basis) 23
   - 9.2 Mapping vs Modelling 23
   - 9.3 Parallel Run and System Coexistence 24
   - 9.4 Transition Principles and Risk Mitigation 25

10. Industry Transition Considerations 25
   - 10.1 Foundation: Clarification and Immediate Guidance 25
   - 10.2 Convergence: Alignment Across Implementations 26
   - 10.3 Target State: Areas for Potential Standardisation 26

11. Conclusion and Key Takeaways 27

Annex 1: Participating IT Providers 29

1.Introduction

This document reflects the outcomes of the IATA IT Providers Sprint Group on Product Catalog, conducted between March and May 2026 under the umbrella of the IATA Modern Airline Retailing Consortium. The initiative brought together multiple IT Providers to consolidate implementation perspectives and contribute to a shared industry understanding of Product Catalog capabilities in the context of Modern Airline Retailing.

The primary objective of the Sprint Group was to examine Product Management capabilities in order to stimulate Product Catalog adoption across Offer and Order stakeholders, accelerate practical industry enablement, and enable consistent processing of Product terminology across the end-to-end lifecycle of airline offers. A secondary objective was to have a shared understanding of the product catalog implementation, strengthening interoperability and supporting coherent transition paths toward Offer and Order transformation.

The scope of this document is limited to industry observations derived from existing Product Catalog implementations. It captures common capabilities, recurring building blocks and data structures, key dependencies and interfaces with adjacent domains, areas of convergence and variation across implementations, and transition considerations from legacy environments. The document focuses on Product Catalog terminology, data structures, modularity, and interactions with adjacent components such as Stock management and Offer Management, while explicitly excluding pricing logic and other commercially sensitive topics, settlement processes, and the definition or modification of IATA standards.

The analysis is intentionally non-prescriptive. It does not define new standards, data schemas, or target architectures, nor does it mandate a specific implementation approach. Instead, it highlights how different solutions address similar functional and integration challenges within heterogeneous technology landscapes, identifying both points of convergence and areas of variation.

The document is structured to progressively present these findings, moving from foundational concepts and terminology to building blocks, dependencies, and interfaces, and culminating in transition considerations and industry alignment perspectives. This progression supports a clear understanding of how Product Catalog capabilities operate across the retailing lifecycle and how they can evolve toward more interoperable implementations.

This document is intended to serve as a reference for airlines and value-chain stakeholders, supporting implementation alignment, interoperability discussions, and informed decision-making. By consolidating current practices and clarifying key dependencies, it aims to facilitate a more consistent and scalable evolution toward Product Catalog capabilities that underpin Modern Airline Retailing.

2.Product Catalog-related Terminology in Use

A consistent and shared terminology is required as a prerequisite to discussing Product Catalog models, dependencies, and interfaces across multiple implementations.

This section describes how Product Catalog concepts are applied in practice across observed implementations. It illustrates how Products are defined, structured, and classified within the Product Catalog, and how these Product terminology elements are subsequently referenced within Offers and Orders.

In line with observed industry practices, terminology usage can be distinguished across three contexts: shared industry use, vendor-specific use, and airline-specific use. These distinctions are reflected in the conceptual illustrations (Ch.5, Fig.3 and Fig.4) through colour coding, where each colour highlights how specific terminology elements are typically used or interpreted across implementations.

Across implementations, a consistent pattern is observed in which the Product Catalog defines Products as structured and reusable entities, independent of commercial constructs such as pricing or availability. A Product is defined as a distinct, atomic item with its own identity, version and attributes, regardless of whether it is sellable as a standalone offering or only in combination with other products. These structural components provide the necessary context for Products to be consistently interpreted and reused across systems. Product terminology elements are then referenced by Offers when presented to customers and by Orders once selected and confirmed, thereby supporting a decoupled model across the retailing lifecycle.

Within this terminology framework, certain concepts are consistently treated as shared industry terminology. In particular, Product Taxonomy and Product Type are used to describe common classification and behavioural structures that support consistent interpretation across systems. Other elements, such as Product Category and Product Category Type, are also used as vendor-specific organisational terminology. In contrast, Product and Product Bundle terminology are typically managed within airline-specific contexts, reflecting how individual airline's structure and group their offerings.

A representative example is observed in the case of seat selection. Within the Product Catalog, a seat is defined as a Product representing a specific seat type, such as an exit row seat. This Product is associated with a Product Type reflecting its functional nature and classified within a Product Category such as ancillary services. Across implementations, variation is observed in how classification is structured, with some IT Providers introducing additional classification layers or organizing Product Types and Product Categories in different hierarchical relationships. A distinction is made between Product Category, which serves a business purpose for grouping and organising products, and Product Type, which defines the technical behaviour of a product, including its data structure, fulfilment logic, and lifecycle characteristics.

This distinction is further clarified, where Product Type represents an atomic, technical classification describing what a Product is and how it behaves, while Product Taxonomy provides a shared industry vocabulary used to classify products at a higher level. Product Category and Product Category Type, by contrast, are primarily used for business organisation and navigation purposes and may vary between implementations.

Within the Product terminology, attributes describe characteristics such as seat location, seat features, and eligibility constraints. These attributes, together with the classification elements associated with the Product, allow it to be consistently interpreted across systems. The Product Catalog maintains this structured terminology independently, while Offers reference the Product when presenting available seat options and Orders reference it once the seat has been selected.

Product Attributes play a central role in this terminology, representing the detailed characteristics of a Product through attribute name-value pairs. These attributes are directly linked to the nature of the Product and to its Product Type, ensuring that both structure and behaviour can be consistently interpreted across systems.

A similar structure is observed for checked baggage. In this case, the Product Catalog defines a baggage as a Product with attributes such as weight limits and size constraints. This Product is also associated with classification elements that position it within broader groupings, such as ancillary services. The combination of attributes and classification provides the necessary structure for consistent reuse across contexts. The Product is defined once and reused across multiple Offers and customer scenarios, while Offers determine how it is presented and under which conditions it is available, and Orders reference the Product for subsequent fulfilment.

It has been observed that some Product Catalog implementations also support the definition of Product Bundles, which represent combinations of multiple Products grouped together under their own Reference (i.e., identifier and version), while preserving in the background references to each individual Product. For example, a bundled proposition may include a seat Product and a baggage Product. In such cases, the Product Catalog maintains the bundle as a structured aggregation of existing Products rather than redefining them. Each Product within the bundle retains its own associated attributes and classification elements, ensuring that the structural context remains intact. This allows Offers to present combined propositions while ensuring that Orders and downstream processes can continue to reference each Product independently.

From a terminology perspective, Product Bundles are typically considered airline-specific terminology constructs, as their composition and structure depend on how individual airlines define and group their Products.

In addition to defining individual Products and bundles, Product Catalog implementations include classification mechanisms that support consistent organization and interpretation.

Product Taxonomies provide structured hierarchies for categorizing Products across domains. For example, a Product such as priority boarding may be classified within a hierarchy that groups it under airport services and further within priority services. These classification structures, alongside Product Types and Product Categories, form complementary terminology layers required for consistent interpretation, enabling filtering, reporting, and interoperability across distribution channels and partners.

Across these examples, a consistent model is observed in which the Product Catalog provides a structured and stable reference for Product terminology, where each Product is described through a combination of attributes and classification elements. Offers and Orders consume this terminology in their respective contexts rather than redefining it. Products are defined once, enriched with contextual information, and reused consistently across the lifecycle, thereby supporting traceability, consistency, and interoperability across implementations.

3. Product Catalog Data Element Concepts and Observed Relationships

To complement the conceptual terminology use presented in the previous section, the table shown in Fig.1 below provides a consolidated view of how participating IT Providers model key Product Catalog data elements and their relationships. It reflects the outcomes of vendor inputs collected during the industry sprint initiative and is intended to illustrate current implementation practices across the various solutions.

Product Catalog Data Element Concepts and Concept Maturity* (Concept ✓ Pilot ✓ Live ✓ Unknown ✓)

* Derived from Solution Maturity

Data Element Concept	May Relate To	A	B	C	D	E	F	G	H	I	J
Product (Product Definition or Description)	Product Attribute (Product Feature, Description Element or Characteristic) (Attribute Key - Value pair)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Product Sub-/Category (Product Taxonomy: airline owned, IATA)	✓	✓	✓	✓		✓	✓	✓	✓	✓
	Product Category Type		✓		✓	✓				✓	✓
	Product Type	✓		✓	✓	✓		✓	✓	✓	✓
Product Category Type	Product Category		✓		✓		✓			✓	✓
Product Attribute	Product Category	✓	✓	✓	✓		✓			✓	✓
	Product Category Type		✓			✓				✓	✓
	Product Type			✓	✓					✓	✓
Product Catalog (Maintenance)	Product	✓	✓	✓	✓					✓	✓
Product Rules	Product	✓	✓	✓	✓					✓	✓
	Product Type	✓		✓	✓	✓				✓	✓

Fig.1

The figure presents a comparative mapping of core data element concepts, the relationships observed between them, and the maturity level at which these concepts are implemented across participating vendors. It is not intended to define a target model or recommend a specific structure, but rather to provide a descriptive overview of how common functional needs are addressed within different solution architectures.

The primary data element concepts identified in the figure represent the core elements used across Product Catalog implementations. These include the concept of Product, representing the definition of a sellable or fulfilment-relevant entity, and Product Attributes, which describe the characteristics associated with that Product. In addition, classification elements such as Product Category and Product Category Type are used to organize and group Products, while Product Type captures the functional nature and behavioural characteristics associated with a Product. The figure also reflects the presence of Product Rules, which define conditions or logic applied either at Product level or across Product Types, as well as Product Catalog maintenance capabilities, which support the management and governance of Product definitions.

Across implementations, these elements collectively form the core building blocks used to structure Product definitions and support their consistent use across Offer and Order processes.

The relationships shown in Fig.1, described under the “May Relate To” column, illustrate how these concepts are connected in practice. These relationships reflect observed modelling approaches rather than mandatory dependencies and should therefore be understood as indicative of how vendors structure their solutions rather than prescribing a specific modelling approach.

In particular, Products are consistently associated with Product Attributes, which provide descriptive characteristics, and with classification elements such as Product Categories or sub-categories, which support organization and grouping. In some implementations, Products are also associated with Product Category Types, introducing an additional classification layer, and with Product Types, which define behavioural semantics. Product Attributes may themselves be linked to Product Types in order to define applicable attribute sets, and in some cases are used as a basis for deriving or assigning Product Categories through business rules. Similarly, Product Rules may be associated either with individual Products, defining conditions at product level, or with Product Types, enabling reuse of rules across multiple Products.

These observed relationships highlight that, while vendors employ different modelling approaches, there is a consistent need to connect the structural, descriptive, and behavioural aspects of Products within the Product Catalog. It is important to note that the presence of a relationship in Fig.1 indicates that it has been explicitly observed in at least one implementation but does not imply that it is universally applied or required.

The figure also provides insight into vendor coverage and the maturity of these concepts. Each column corresponds to a participating IT Provider, and the presence of a checkmark indicates that the respective concept or relationship is supported within that implementation. The colour coding reflects the maturity level reported by vendors, distinguishing between concepts that are defined at a conceptual level, those that are being piloted, and those that are already deployed in live production environments, as well as cases where maturity information was not specified.

This maturity view provides additional context on how broadly established each concept is across the current landscape. It illustrates that while some capabilities are widely implemented and operational, others are still evolving or being adopted progressively across the industry.

The analysis of vendor responses further highlights a number of consistent patterns emerging across implementations. These observations confirm a broad alignment on the fundamental concepts required to support Product Catalog capabilities, despite differences in modelling approaches and system architectures. Core product constructs are consistently present, with Products defined as structured entities combining attributes, behavioural elements, and classification mechanisms. While the specific organisation of these elements varies, they fulfil comparable roles in enabling consistent product definition, interpretation, and reuse across systems. At the same time, variability remains in areas such as classification structures and the implementation of rules and governance mechanisms, reflecting different approaches to organising product information and managing business logic within Product Catalog solutions.

Overall, Fig.1 should be interpreted as a descriptive consolidation of current practices across participating IT Providers. It does not prescribe a standard data model or suggest a preferred implementation approach. Rather, it provides a structured overview of commonly used data element concepts, illustrates how these concepts are interrelated in practice, and offers insight into their maturity across the industry. This consolidated view serves as a foundation for the subsequent analysis of common building blocks, key dependencies, and transition considerations associated with Product Catalog capabilities.

4. Product Catalog Interactions and Observed Responsibilities

To complement the description of Product Catalog data elements and their relationships Fig. 2 below provides a consolidated view of how Product Catalog capabilities interact with adjacent systems across the airline retailing ecosystem, as reported by participating IT Providers.

Product Catalog Interactions* and Solution Maturity (Concept ✓ **Pilot ✓** **Live ✓** **Unknown ✓**)

* Interactions indicate data or functional integration; they do not imply ownership or operational responsibility.

Actor/System	Direction	Interaction	A	B	C	D	E	F	G	H	I	J
User Interface	Source → Catalog	Create / Modify Product	✓	✓	✓	✓	✓				✓	✓
	Consumer ← Catalog	Search / Get Product	✓	✓	✓	✓	✓				✓	✓
Airline** / Partner Product Catalog	Source → Catalog	Create / Modify Product(s)	✓	✓	✓	✓	✓		✓	✓	✓	✓
	Consumer ← Catalog	Get Product(s)	✓	✓	✓	✓	✓		✓		✓	✓
Schedules	Source → Catalog	Create / Modify Transport Product (Flight, Orig., Dest., Date)	✓	✓	✓	✓	✓		✓	✓	✓	✓
Stock Keeper	Consumer ← Catalog	Use Product data to manage quantity and availability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Offer Management	Consumer ← Catalog	Search Product(s)	✓	✓	✓	✓	✓	✓	✓	✓		✓
	Consumer ← Catalog	Return Eligible Product(s)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Order Management	Consumer ← Catalog	Use Product Reference	✓	✓	✓	✓		✓	✓	✓	✓	✓
Order Accounting	Consumer ← Catalog	Use Product Reference	✓	✓	✓	✓		✓	✓	✓	✓	✓
Delivery with Orders	Consumer ← Catalog	Use Product Reference	✓		✓	✓		✓	✓	✓	✓	✓
Seller	Consumer ← Catalog	Send Airline Profile (Flight Orig., Dest., Frequency)	✓	✓	✓	✓	✓		✓	✓	✓	✓
Interline Partner	Consumer ← Catalog	Provide Supplier Product	✓	✓	✓	✓	✓				✓	

** Airline in either Retailer or Supplier role

Fig.2

The figure is structured as a matrix showing, for each actor or system, the direction of interaction with the Product Catalog and the type of interaction performed. It also indicates the level of implementation maturity of these interactions across different vendor solutions. As indicated in the figure, interactions represent data or functional integration points and do not imply ownership or operational responsibility.

A first set of interactions relates to user-driven access to the Product Catalog. The User Interface is shown to interact with the Product Catalog both as a source and as a consumer. In several implementations, the User Interface can create or modify product definitions, reflecting administrative or configuration capabilities exposed to business users. More consistently across vendors, the User Interface consumes Product Catalog services to search for and retrieve Product definitions, supporting visualization and management of catalog content.

The figure further shows that Airline or Partner Product Catalogs act both as sources and consumers of Product Catalog data. In some implementations, airline or partner systems create or update product definitions within the Product Catalog, reflecting decentralised or federated product management models. At the same time, these systems may retrieve product definitions for synchronization or operational purposes, highlighting the need for consistent product information exchange across organizational boundaries.

A specific interaction is observed with Schedules systems, which in several solutions provide input for creating or modifying transport-related products, such as flights. These interactions typically include information such as origin, destination, and date, enabling the Product

Catalog to define transport products based on schedule data. This illustrates how certain product types are derived from operational sources rather than manually defined.

Stock Keeper systems interact with the Product Catalog by consuming product data to manage quantities and availability. This interaction is consistently observed across implementations and reinforces the separation between product definition, which resides in the Product Catalog, and inventory management, which is handled by Stock Keeper components. Products are therefore defined independently of availability, while Stock Keeper systems apply capacity and allocation constraints.

Offer Management components are shown to consume Product Catalog capabilities through two primary interactions. First, they search for available Products within the Product Catalog, and second, they retrieve eligible Products based on defined criteria. These interactions reflect the role of the Product Catalog as a source of product definitions used in constructing Offers. Offer Management determines how Products are combined, priced, and contextualized, but relies on the Product Catalog for consistent product information.

Order Management systems are shown to interact with the Product Catalog by using Product references. Rather than redefining product information, Orders rely on references to Product definitions maintained within the Product Catalog. This pattern is consistently observed across vendors and enables traceability and consistency between the Offer and Order stages of the retailing lifecycle.

Similar interactions are observed for Order Accounting and Delivery functions. Both components consume Product references to support their respective processes. In the case of accounting, product references are used to allocate revenue and ensure financial traceability. For delivery and fulfilment, product definitions, and in particular Product Type and associated attributes, enable the correct execution of the service. These interactions highlight the importance of maintaining stable product identifiers and consistent definitions across downstream systems.

Seller systems are also shown to interact with the Product Catalog by receiving airline profile information, such as origin, destination, and frequency data. This interaction reflects how contextual and operational information provided by sellers can influence product availability or eligibility within the overall retailing flow.

Finally, interactions with interline partners are observed, where partner systems provide supplier product information. This reflects the need for Product Catalog capabilities to support cross-organization scenarios, in which products defined by one party must be interpreted and used by another. These interactions further reinforce the importance of consistent product definitions and stable referencing mechanisms across organizational boundaries.

Across all these interactions, a consistent pattern emerges in which the Product Catalog acts as a provider of product definitions that are consumed by multiple downstream components.

While some systems contribute to product definition, the majority of interactions involve consuming or referencing Product Catalog data. This confirms the central role of the Product Catalog as a shared reference layer within the retailing ecosystem.

The maturity indicators included in the figure show that these interaction patterns are already implemented across several vendor solutions, although with varying levels of deployment. Core interactions, such as product search, retrieval, and reference in Orders, are widely available and often implemented in live environments. Other interactions, particularly those involving cross-system integration or partner exchange, exhibit more variability, reflecting differences in implementation roadmaps and architectural approaches.

Overall, Fig. 2 illustrates a consistent set of interaction patterns that position the Product Catalog as a central, shared capability supporting multiple domains across the retailing lifecycle. At the same time, it reinforces the importance of clearly separating responsibilities between systems, ensuring that product definition, commercial logic, and operational execution remain distinct while interoperating through well-defined interfaces.

5. Conceptual Models of Product Catalog Implementations

Building on the analysis of Product Catalog data elements and their interactions with adjacent systems, the Sprint Group examined how these elements are combined into different conceptual models across vendor implementations. The objective of this analysis was not to classify solutions into fixed categories, but to understand how similar functional capabilities can be achieved through different modelling approaches.

The conceptual approaches identified in this analysis are illustrated in Fig. 3 and Fig. 4 below, which provide representative views of how Product Catalog building blocks can be organised within different modelling frameworks. These figures highlight common patterns observed across implementations and, in particular, different ways of representing composite products within the Product Catalog.

As introduced in the terminology section, these conceptual models also reflect the different usage contexts of Product Catalog elements, illustrated through colour coding. Shared industry constructs (e.g. Product Type and Product Taxonomy), vendor-specific organisational elements (e.g. Product Category), and airline-specific constructs (e.g. Product and Product Bundle) are represented distinctly, highlighting how responsibilities and interpretations vary across implementations.

A first key observation is that all conceptual models share a consistent foundation, in which Products are defined as structured entities described through attributes and associated with behavioural characteristics. Across all implementations, Products remain the fundamental unit used to represent what is offered, ordered, and fulfilled within the retailing flow.

Existing Product Catalog Conceptual Model (with "seat" as example)

Relationships indicate this group's observed patterns; they do not imply mandatory implementation or ownership.
Object multiplicity (i.e., many-to-many, zero-to-many, etc.) is available upon request.

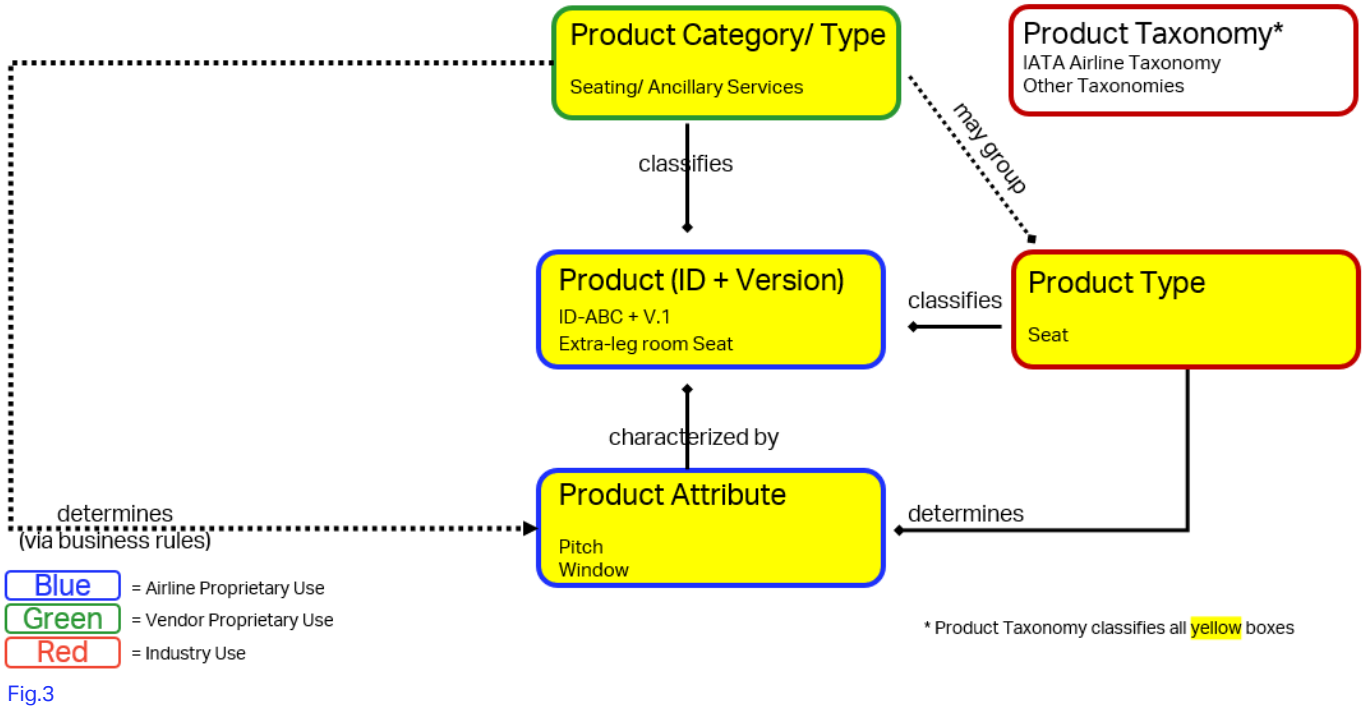

Fig.3

The primary distinction between the models illustrated in Fig. 3 and Fig. 4 lies in the treatment of composite offers, and more specifically in whether bundles are represented as Products within the Product Catalog or handled purely as combinations of individual Products.

Fig. 3 illustrates a model in which the Product Catalog contains only atomic or component-level Products, and where composite propositions are not modelled as standalone Product entities. In this approach, what is commercially offered as a bundle is constructed outside the Product Catalog, typically within Offer management, by combining multiple individual Products. The Product Catalog therefore remains focused on defining reusable product components, each with its own attributes, classification, and behaviour. Bundling is treated as a commercial construct rather than a structural element of the Product Catalog.

In contrast, Fig. 4 presents a model in which bundles are defined as first-class entities within the Product Catalog itself. In this approach, a bundle is represented as a Product with its own identifier, attributes, and lifecycle, in addition to maintaining references to the individual Products that compose it. This allows the Product Catalog to explicitly represent not only atomic products, but also commonly offered combinations of products as structured entities. As a result, product bundles can be directly defined and managed within the Product Catalog, rather than being constructed dynamically in downstream components.

Existing Product Catalog Conceptual Model (with "seat" as example)
Relationships indicate this group's observed patterns; they do not imply mandatory implementation or ownership.
Object multiplicity (i.e., many-to-many, zero-to-many, etc.) is available upon request.

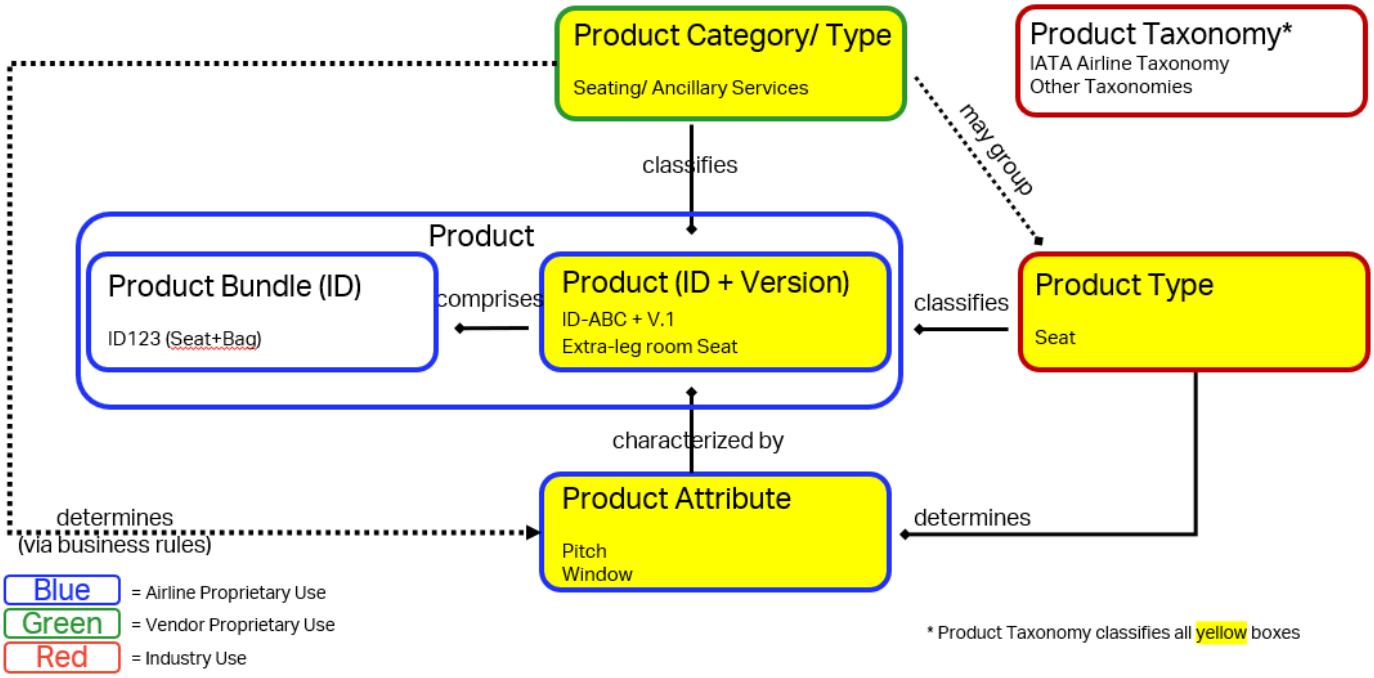

Fig.4

Across both models, the relationships illustrated between elements represent observed patterns rather than prescribed structures. In particular, relationships between Products, Product Types, Taxonomies, and classification elements may vary in cardinality (e.g. one-to-many, many-to-many, or optional relationships), depending on implementation choices. These variations do not imply different functional outcomes but reflect the flexibility available in how systems structure and associate Product Catalog components.

Despite this difference, a consistent pattern is observed across both models: the underlying product components remain individually defined and retain their own attributes, classification, and behavioural semantics. Whether or not a bundle is treated as a Product, downstream processes such as fulfilment, servicing, and accounting continue to rely on the individual Product definitions. This ensures that operational processes can consistently interpret and execute each component of a bundle regardless of how it is represented at the catalog level.

Additional variations exist between implementations, including the way Products are classified or how eligibility rules are associated with Products or Product Types. However, these differences are secondary compared to the distinction in how bundles are represented. In both models, classification mechanisms such as Product Category and Product Taxonomy serve to organise and contextualise Products, and Product Types or equivalent constructs define what the product is. As reflected in the colour-coded representations, these elements may belong to different usage contexts (industry, vendor, or airline), further influencing how they are structured and applied across implementations.

Importantly, the analysis shows that the different treatments of bundles do not prevent interoperability in principle. Regardless of whether bundles are modelled as Products or constructed outside the product catalog, implementations must be able to expose Product definitions, maintain stable product references, and ensure that underlying product components can be consistently interpreted across systems. This reinforces the importance of focusing on how Products are referenced and understood, rather than how they are internally organised within the Product Catalog.

In summary, the conceptual models illustrated in Fig. 3 and Fig. 4 can therefore be understood as alternative ways of organising the same underlying building blocks, with the main distinction being the role of bundles within the Product Catalog. One approach emphasises the Product Catalog as a repository of atomic products, with bundling handled externally, while the other incorporates bundles directly as managed catalog entities. In practice, some implementations adopt hybrid approaches, where certain bundles are defined within the Product Catalog while others are constructed dynamically based on context.

Each of these approaches presents advantages and trade-offs. Models that limit the Product Catalog to atomic products provide clear separation between product definition and commercial composition, which can simplify governance and increase flexibility in offer construction. Models that include bundles as Products enable reuse of predefined product combinations and can simplify the management of frequently offered packages but may introduce additional complexity in maintaining consistency between bundle definitions and their components.

The analysis confirms that no single approach can be considered superior, as each reflects different implementation contexts and design priorities. Instead, the coexistence of these models highlights the importance of ensuring that, regardless of structure, Product definitions and their components can be consistently referenced, interpreted, and executed across the retailing lifecycle.

From an industry perspective, this finding reinforces that alignment efforts should focus on the consistent handling and interpretation of Products and their components, rather than enforcing a specific approach to modelling bundles. As long as underlying Product definitions remain stable and interoperable, different approaches to bundle representation can coexist without limiting the ability to support end-to-end retailing processes.

The comparison of these approaches, as illustrated in Fig. 3 and Fig. 4, provides a foundation for the subsequent analysis of key dependencies, where the focus shifts from internal modelling choices to the conditions required to ensure consistent behaviour and interoperability across systems and organisational boundaries.

6. Common Building Blocks of Product Catalog Implementations

The analysis of Product Catalog implementations across participants resulted in the identification of a set of common building blocks, evaluated based on their impact on interoperability, whether they are considered mandatory, and the conditions under which flexibility is acceptable. Rather than describing a conceptual structure, this section reflects a consolidated assessment of these elements, focusing on their role in enabling consistent product understanding and exchange across systems.

A first group of building blocks is consistently identified as both high impact and mandatory. Product Definition, including core attributes (i.e., Product Reference (ID + version as explained below) and Product Attributes), is recognised as a fundamental element of the Product Catalog required to support product understanding and exchange across systems. While essential, its scope is explicitly limited to static characteristics and must not include pricing or entitlement logic.

Similarly, Product Taxonomy and Category are considered mandatory due to their role in enabling shared understanding and interoperability across vendors. However, while the concept must be commonly understood, flexibility is allowed in how classification structures are implemented, provided that mappings exist where needed.

Product Reference, including identifier and versioning, is also identified as a critical and mandatory element. It ensures traceability, exchange, and lifecycle consistency across systems. A further requirement is the stability of this Product Reference across airline domains and over time, which is necessary to maintain consistency across Offer, Order, Delivery, and reporting processes. This stability must persist even for discontinued products, while interim identifier schemes may be used if they remain forward-compatible.

A second set of building blocks relates to rule management, where a clear distinction is made between what belongs inside and outside the Product Catalog. Static Product Rules are considered mandatory with medium impact, as they ensure correct applicability of Products without requiring runtime logic. Their use is however limited to static eligibility conditions and is only required when these rules must be shared across systems.

In contrast, Dynamic Rules and Pricing Logic are explicitly identified as out of scope for the Product Catalog. These elements belong to Offer Management and must not be embedded in Product Catalog modelling, in order to preserve the separation between product definition and commercial logic.

Transition-related building blocks are also identified, reflecting the realities of existing system landscapes. When applicable, for products that are sold or have to be represented in the legacy systems, legacy Mapping (including constructs such as SSRs and RFISCs), is

considered necessary but only during transition phases. Its role is limited to ensuring short-term operational continuity and interoperability with legacy systems. Importantly, these mappings must remain output-only and must not drive Product semantics, ensuring that legacy constructs do not influence the Product model.

Similarly, the Translation Layer between legacy and modern representations is required during transition to support system coexistence. However, it is explicitly treated as a temporary capability, expected to follow a defined sunset path as transformation toward Offers & Orders matures.

A third set of building blocks focuses on architectural alignment and system boundaries.

Product–Offer–Order clear boundaries are also considered mandatory and high impact, as they prevent semantic duplication and architectural drift. These boundaries must be clearly defined through contractual interfaces between systems.

Stock and availability associations are recognised as necessary for offer feasibility but are not considered mandatory within the Product Catalog itself. These elements must remain external to the Product definition, with ownership and responsibility explicitly defined.

Operational transition considerations are further reflected through the Parallel Run Coexistence Model, which is required for transitioning airlines. This model ensures business continuity during migration and favours incremental rollout approaches over large-scale “big bang” transitions.

Finally, governance-related building blocks define what should explicitly not be modelled within the Product Catalog. Governance on what not to model is identified as a high-impact and mandatory practice to prevent long-term lock-in into legacy constructs. This governance must be documented explicitly and consistently applied.

Predefined Core Products, representing industry baselines, are identified as optional but valuable elements. While not mandatory, they can accelerate onboarding and interoperability by providing a limited set of commonly defined Products with agreed attributes.

Across these evaluated building blocks, a clear pattern emerges. A subset of elements, including Product Definition, Product Reference, Product Taxonomy, and system boundary definition, is consistently identified as essential for interoperability and must be implemented across all solutions. Other elements, such as classification structures, bundles, or predefined product sets, allow flexibility, provided that core interoperability requirements are preserved.

Transition-specific elements, including legacy mapping and translation layers, are required only temporarily and must be managed in a controlled manner to avoid introducing long-term dependencies.

This evaluation confirms that interoperability is not driven by uniform modelling approaches, but by the consistent implementation of a defined set of high-impact building blocks, combined with clear separation of concerns and controlled flexibility where appropriate.

7. Key Dependencies Between Product Catalog Building Blocks

Following the identification of common building blocks, the analysis focuses on the dependencies between these elements. These dependencies are evaluated based on their type (e.g. structural, data, or reference), the reason they exist, the operational risks if misaligned, their criticality level, and the degree of flexibility allowed. This section reflects a consolidated view of the dependency pairs identified across implementations.

At the core of the dependency model are reference-based dependencies, which are consistently identified as must-have for interoperability. A foundational dependency exists between Product and Product Reference (ID and version). This is a reference dependency, as the identifier is required to uniquely identify the Product across systems. If product reference is not stable, Products cannot be consistently identified, leading to downstream inconsistencies across Order, Delivery, and Accounting processes. This dependency is categorised as must-have, with flexibility limited to how versioning is implemented.

A closely related dependency concerns the stability of Product Reference over time. This ensures that the same Product identifier persists across the lifecycle. If not maintained, systems may reference outdated or incorrect versions of a Product, resulting in inconsistencies in downstream processing. This dependency is considered recommended, allowing flexibility in version evolution, provided backward traceability is maintained.

This extends to a dependency on Product Reference stability across systems, where the same Product must be consistently referenced in multiple systems. This is also a reference dependency and is considered must-have. If misaligned, cross-system inconsistencies arise, particularly affecting Order processing and downstream operations. Flexibility is allowed through mapping mechanisms when identifiers differ between systems.

In interline scenarios, additional dependencies are introduced. A dependency exists between interline partners and stable Product references, where partners rely on consistent identifiers. If not maintained, interline processes fail. The dependency level is context-dependent, typically must-have or recommended depending on the use case. Similarly, a dependency may exist between interline partners and identifier mapping, which may be required when supplier and retailer identifiers differ. Without mapping, Products may not be recognised across organisations. Mapping may be required, although its implementation may remain internal and not necessarily exposed to partners.

Beyond reference dependencies, structural dependencies define how Products are interpreted. A dependency exists between Product and Product Type, where Product Type classifies the Product. This structural dependency is flexible, as different implementations may adopt different modelling approaches. If misaligned, interoperability may be limited, although this can be mitigated through taxonomy alignment or mapping.

Further dependency exists between Product and Product Attributes. This is a data dependency within the Product Catalog, as attributes define the characteristics of a Product. This dependency is considered must-have to ensure a complete and valid Product definition. However, the dependency remains internal to the Product Catalog and does not require consistency across systems or partners, which rely on the resulting Product representation rather than on the underlying attribute structures.

Dependencies also exist between Product and Product Taxonomy, and between Product Type and shared taxonomy structures. These are structural dependencies within the Product Catalog, supporting classification and grouping. Inconsistent taxonomy associations may lead to incorrect classification and impact filtering, reporting, or navigation. These dependencies are therefore important for ensuring coherence within the Product Catalog, but their implementation does not require alignment across systems or partners, which rely on the resulting classification and common interpretation of the taxonomy rather than the underlying implementation of the taxonomy structure.

A dependency exists between Product Bundles and individual Products, whereby bundles reference underlying atomic Products through their Product References (ID + version). This ensures that each component within the bundle is uniquely identifiable and can be resolved for fulfilment and downstream processing. If this dependency is not maintained, bundles become ambiguous and cannot be decomposed into their constituent Products, preventing consistent interpretation and execution. This dependency is therefore classified as must-have, with no flexibility on the requirement for bundle decomposition and clear identification of its components.

Dependencies across the Product–Offer–Order lifecycle are also explicitly identified. Offer and Service components depend on Product References in order to reference Products during Offer creation. Without this dependency, Offers cannot link to Products. This is considered a must-have reference dependency.

Similarly, Order and Service components depend on Product References to ensure traceability. If this dependency is not maintained, Orders cannot be traced back to Products, leading to failures in servicing and reporting. This dependency is also must-have.

Delivery processes depend on Products indirectly through Orders, rather than through direct Product dependencies. This dependency is considered recommended, with the design principle that Delivery should rely on Order information rather than direct Product coupling.

Order Accounting introduces a dependency on Order and Service references, as accounting relies on traceability through Orders. If misaligned, accounting issues may arise. This dependency is must-have.

Transition-specific dependencies are also identified. A dependency exists between Product representations and legacy codes, including RFISCs and SSR.s These are reference-based dependencies required for legacy support during transition. If not maintained, legacy processes break. However, RFISC and SSR are considered must-have during transition, while Fare Basis is now treated as optional or "nice-to-have", primarily for optimisation purposes rather than mandatory interoperability. These dependencies should be implemented with minimal or subset mapping and must not influence Product semantics.

Finally, a dependency exists between parallel run environments and stable Product References. During transition, legacy and modern systems must coexist. If Product references are not stable, inconsistencies and drift occur between systems. This dependency is considered must-have, with transition-specific mechanisms required to support coexistence.

Across all identified dependencies, a consistent pattern emerges. Reference dependencies, particularly those involving Product identifiers and lifecycle stability, are consistently classified as must-have, as they directly impact core processes across Offer, Order, Delivery, and Accounting. Structural and semantic dependencies, such as Product Type or Taxonomy, allow more flexibility, provided that alignment mechanisms or mappings are in place.

This distinction between mandatory dependencies and areas of controlled flexibility enables interoperability without requiring uniform modelling approaches. It ensures that systems can differ in how they structure and organise Product Catalog elements, while still maintaining consistent behaviour across the end-to-end retailing lifecycle.

8. Interfaces and Information Exchange Between Product Catalog and Other Systems

The analysis of Product Catalog implementations identifies a consistent set of interfaces that must exist between the Product Catalog and other components of the retailing ecosystem. These interfaces define how Product information is accessed, exchanged, and used across internal airlines systems such as Offer Management, Order Management, Stock management, and downstream processes, and partners.

This analysis is structured around three key dimensions: the minimum information that must exist across interfaces, the elements that should remain vendor-specific, and the level of detail required to support interoperability.

8.1 Minimum Product Information Across Interfaces

A first key requirement is the definition of a minimum set of Product information that must be consistently available across all interfaces. This minimum standardised layer consists of Product Reference (identifier and version), Product Taxonomy, and core Product Attributes.

These elements form the smallest shared representation of a Product that allows it to be consistently identified and interpreted across systems. Without this baseline, interoperability between systems cannot be ensured.

This minimum layer must be applied consistently across modules through a reference-based approach, where Product data is not duplicated but always accessed via the Product Catalog as the source of truth. Offer Management systems require full access to Product definitions, including attributes, bundle composition (when applicable), and constraints. Order Management systems rely on consistent Product Reference (ID and version) to maintain a stable definition of what has been purchased. Stock management systems equally depend on consistent Product Reference and attributes for capacity management.

A consistent set of interaction patterns is observed across implementations:

- Product Catalog - Offer: full Product definition access (including bundles when applicable and constraints)
- Product Catalog - Order: Consistent product Reference (ID + version) for definition lookup
- Product Catalog - Stock: Consistent product Reference (ID + version) and attributes for capacity handling

8.2 Vendor-Specific and Context-Dependent Elements

The analysis clearly distinguishes between elements that must be standardised and those that should remain vendor-specific. Certain elements are explicitly identified as vendor-specific and outside the scope of standardised Product Catalog interfaces. These include but are not limited to:

- Scheduling integration and operational system interfaces
- User interface layers
- Workflow and optimisation logic, including internal orchestration and tooling

In addition, several elements are implicitly vendor-specific, as they relate to commercial or system-specific logic rather than Product definition itself. These include but are not limited to:

- Pricing implementation and pricing structures
- Offer-level rules such as personalisation, segmentation, and eligibility logic
- Offer engine internal processing and decision logic
- Airline-specific negotiation logic
- Rich content, including media, URLs, and extended descriptions

These elements are not part of the Product Catalog interface and are therefore not required to be shared in a standardised way. This separation ensures that Product definition remains distinct from commercial logic and system-specific implementation details.

8.3 Level of Detail for Interoperable Exchange

A further distinction is made regarding the level of detail that must be exchanged across interfaces. A minimum standardised layer must always be shared, consisting of Product Reference (ID and version), Product Taxonomy, and core Product Attributes. This layer ensures consistent product identification and interpretation across systems.

Beyond this minimum layer, additional information may be exchanged but is not consistently standardised. These extended elements include but are not limited to:

- Detailed or extended attributes
- Product Rules
- Pricing logic and conditions
- Rich content (shared via references such as links or URLs rather than duplication)
- Offer-level rules and decision logic
- Context-dependent enrichment (e.g. schedule attributes, taxonomy metadata, airline profile filters)

The level of detail shared depends on the context, the systems involved, and the degree of integration between partners.

8.4 Overall Interface Principles

Across all implementations, a consistent set of principles emerges. Interoperability does not require full standardisation of all Product information, but rather the consistent use of a limited shared data layer combined with reference-based access patterns and clear separation of concerns.

Product Catalog acts as the authoritative source for Product terminology, while consuming systems access and interpret this information through well-defined interfaces without duplicating data. At the same time, vendor-specific logic and extended information remain outside of the standardised interface layer, enabling differentiation without compromising interoperability.

This approach allows heterogeneous implementations to coexist while ensuring that Products can be consistently identified, referenced, and used across the end-to-end retailing ecosystem.

9. Transition Considerations for Product Catalog Implementations

The analysis of Product Catalog implementations highlights that transition from legacy environments is a critical aspect of solution design. This transition must be managed explicitly through a combination of required capabilities, avoided practices, and elements that should be progressively de-prioritised over time.

Three main areas structure this analysis: the treatment of legacy constructs, the distinction between mapping and modelling, and the realities of parallel system coexistence. Each area is evaluated in terms of what is required during transition, what is not recommended, and what should be reduced over time.

9.1 Legacy Constructs (RFISC, SSR, Fare Basis)

During transition, legacy constructs such as RFISCs and SSRs remain necessary in specific operational contexts, including interline, settlement, and compatibility with existing systems. These constructs must therefore be supported where operationally unavoidable.

At the same time, legacy codes must be treated strictly as outputs of the Product Catalog and must not serve as inputs defining Product semantics. Product terminology must be defined independently, with legacy representations derived through controlled mechanisms.

Different legacy constructs require differentiated treatment. SSR and RFISC codes are often unavoidable due to operational dependencies. but can be partially reduced over time. In contrast, Fare Basis is no longer required for modern Product Catalog implementations and should be treated as optional or “nice-to-have”, primarily for Offer purposes rather than core interoperability.

To support this approach, controlled transition layers must be implemented to preserve compatibility without impacting the end-state Product model.

Over time, reliance on legacy constructs must be progressively reduced. In particular, dependency on SSR/RFISC semantics, continued usage of Fare Basis as a product construct, and long-term reliance on translation layers without a defined retirement path should be de-prioritised.

9.2 Mapping vs. Modelling

A second area of focus concerns the distinction between mapping and modelling. Mapping is required during transition to ensure interoperability with existing systems, including standard codes and operational data required for delivery, servicing, accounting, and settlement. This includes mapping:

- legacy codes such as SSR and RFISC

- operational attributes such as cabin information, seat characteristics, and origin - destination data

At the same time, a strict separation must be maintained between mapping and modelling. Mapping supports interoperability, whereas modelling defines the Product Catalog itself.

Practices that are not recommended include embedding legacy constructs directly into Product models, incorporating airline-specific legacy logic that does not generalise, or introducing transitional workarounds designed only to accelerate early migration without long-term viability.

Over time, long-term reliance on mapping layers must be reduced. In particular, continued dependence on mapping without convergence toward native Product Catalog representations, and the use of legacy-driven representations beyond interoperability needs, should be progressively de-prioritised.

9.3 Parallel Run and Coexistence

The transition toward modern Product Catalog capabilities requires coexistence between legacy and modern systems for an extended period. As a result, explicit support for parallel run scenarios is required.

This includes:

- controlled transition layers ensuring interoperability between systems
- process orchestration and mapping across legacy and modern environments
- consistent Product identifiers across Product, Offer, Order, and Stock systems
- stability of operational processes such as delivery, servicing, and settlement

Clear separation of responsibilities must be maintained between product definition, availability, pricing, and fulfilment, ensuring that each function interacts with the Product Catalog in a controlled and consistent manner.

Certain practices are explicitly not recommended. These include duplication of semantics between legacy and modern models, maintaining parallel workflows without a convergence strategy, and recreating legacy logic within modern systems. These approaches introduce unnecessary complexity and increase the risk of long-term inconsistency.

Over time, transition-specific mechanisms must be reduced. In particular, permanent reliance on translation layers, legacy-driven orchestration, workarounds maintained solely for backward compatibility, and continued dependence on legacy constructs for analytics or product definition should be de-prioritised.

9.4 Overall Transition Principles

Across these areas, a consistent pattern emerges. Transition is not simply a technical migration but a controlled process requiring clear separation between interim capabilities and target-state design.

Successful transition requires:

- controlled use of legacy constructs as outputs only
- clear distinction between mapping and modelling
- stable Product references across systems
- explicit coexistence strategies with defined retirement paths

At the same time, practices that introduce long-term dependency on legacy semantics, mapping layers, or duplicated logic must be avoided or progressively eliminated.

This approach ensures that legacy compatibility can be maintained during transition, while enabling a gradual evolution toward a clean, interoperable, and product-centric retailing model.

10. Industry Transition Considerations

Building on the analysis of Product Catalog building blocks, dependencies, interfaces, and transition considerations, a set of practical recommendations emerges to support the industry in progressing toward interoperable Product Catalog implementations.

These recommendations are structured across three transition stages: establishing a clear foundation, driving convergence across implementations, and progressing toward a stable target state. Each stage reflects a different level of maturity and alignment required across the industry.

10.1 Foundation – Clarification and Guidance

In the initial stage of transition, the priority is to reduce ambiguity and establish a clear and shared understanding of how Product Catalog capabilities should be applied. This stage focuses on clarifying roles, boundaries, and minimum expectations without introducing new standards.

A core priority is to define clear boundaries between Product Catalog and Offer logic, including explicit separation of responsibilities for pricing, segmentation, and personalisation, which must remain outside the Product Catalog.

Legacy constructs require explicit clarification in this phase. Their handling must follow clear principles: legacy codes are output-only, transitional, and time-bounded. Product semantics must not be derived from constructs such as RFISC, SSR, or Fare Basis.

A minimum set of Product data elements must also be clearly defined and consistently applied across implementations. This includes agreement on Product identifiers, versioning, and core attributes required by downstream systems such as Order, Delivery, and Accounting.

Further clarification is required on the role and scope of Product Taxonomy. Specifically, taxonomy must be used for classification and grouping but must not be overloaded with Product behaviour or business logic.

Finally, early-adopter implementation practices should be captured and shared as reference guidance, allowing the industry to build on proven approaches without requiring immediate standardisation.

10.2 Convergence – Alignment Across Implementations

The second stage focuses on areas where alignment across implementations is both achievable and beneficial. The objective is to reduce fragmentation by converging on common approaches while still allowing flexibility in implementation design.

A key area for convergence is the establishment of a common baseline for Product Taxonomy. Alignment on Taxonomy structures and interpretation reduces divergence and improves interoperability across vendors and partners.

Alignment is also required for Product identification which should be globally unique and have consistent versioning mechanisms to ensure traceability across systems and over time.

Interaction patterns across modules must also be harmonised. This includes alignment on how Product information is accessed and exchanged between Product Catalog, Offer, Order, and supporting systems, as well as alignment across airline, retailer, and partner roles.

Finally, the industry should reduce reliance on bespoke vendor-specific mappings. Where possible, shared implementation practices should replace one-off integrations, improving consistency and reducing complexity across systems.

10.3 Target State – Potential Areas for Standardisation

The final stage reflects the target state of the transition, where deeper alignment may require formal standardisation in selected areas to support full interoperability and ecosystem integration.

An objective that would be “nice-to-have” is the definition of consistent approaches for translating between legacy and modern representations. This includes standardised handling of legacy constructs to ensure compatibility while enabling full transition to modern Product models.

Standardisation of cross-module interaction and data exchange is also required. This includes defining consistent interface patterns and data contracts across Product Catalog, Offer, Order, and downstream systems.

Governance of Product definitions represents another critical area. This includes harmonised management of Product identity, versioning, and lifecycle, ensuring that Products remain consistently traceable and interpretable across systems.

Additional areas for alignment include shared data models for classification and contextual information. This includes taxonomy structures, airline profiles, and supplier catalog models, which should evolve toward more consistent industry usage.

Security and data protection also become increasingly important. Standardised approaches are required to ensure trust, compliance, and consistent handling of Product data across systems and organisations.

11. Conclusion and Key Takeaways

This document has consolidated observations from participating IT Providers to provide a structured and industry-aligned view of Product Catalog capabilities within the context of Modern Airline Retailing. Across diverse implementations, a consistent picture emerges: while architectural approaches and conceptual models differ, there is strong convergence on the fundamental elements required to support product-centric retailing.

At the core of this convergence is the recognition that Products must be defined as structured, reusable entities with stable identifiers, consistent attributes, and clearly defined behavioural semantics. In particular, the use of stable Product References (ID + version), aligned Product Attributes, and consistent Product Type semantics forms the foundation upon which Offers, Orders, and downstream processes operate. Regardless of whether implementations are type-driven, attribute-driven, or adopt hybrid approaches, interoperability depends on the consistent interpretation of these core Product definitions rather than on alignment of internal models.

A second key observation is that the Product Catalog acts as a central reference capability within the retailing ecosystem. It provides the authoritative definition of Products, while other components consume these definitions through well-defined interfaces. This reinforces a clear separation of concerns: the Product Catalog defines what a Product is, while Offer Management determines how it is presented and priced, and Order and downstream systems ensure its execution. Maintaining this separation is critical to enabling modularity, scalability, and consistent behaviour across systems.

The analysis also highlights that differences between implementations are most visible in how Products are structured and combined, particularly in the treatment of bundles. Some implementations represent bundles as explicit Product entities, while others construct them

dynamically from individual Products. Despite this variation, a consistent requirement exists across all models: any composite must ultimately resolve to clearly defined Product components in order to support fulfilment, servicing, and accounting processes.

Building on this foundation, the identification of common building blocks and their dependencies provides a more precise understanding of where alignment is required. Dependencies related to Product Reference, Product Type semantics, attribute consistency, and bundle resolution are consistently identified as high-impact and non-negotiable. Misalignment in these areas leads directly to failures in core processes across the retailing lifecycle. Conversely, other areas, including classification structures, attribute modelling approaches, and bundle representation, allow flexibility, provided that these must-have dependencies are preserved and supported through mapping or alignment mechanisms where required.

The analysis of interfaces reinforces the importance of a shared minimum data layer. Interoperability does not require full standardisation of all Product information, but it does require consistent exchange of key elements: Product References (ID + version), Product Taxonomy, and core Product Attributes. This minimum standardised layer enables consistent identification and interpretation across systems, while extended data and vendor-specific logic remain outside the interface scope. At the same time, a clear boundary must be maintained between shared Product data and vendor-specific logic, including pricing, personalisation, and internal processing. enabling both interoperability and differentiation.

Transition considerations confirm that the move toward modern Product Catalog capabilities must be managed progressively. Legacy constructs remain necessary in many operational contexts but must be treated as output-only, transitional representations derived from Product definitions and must not drive Product semantics. The use of mapping layers, controlled coexistence mechanisms, and stable Product References allows legacy and modern systems to operate in parallel while ensuring that long-term dependency on legacy constructs, translation layers, and duplicated logic is progressively reduced.

From these observations, a consistent set of industry priorities emerges. In the foundation phase, clarity is required on roles, boundaries, and minimum data requirements. In the convergence phase, alignment is needed on key elements such as taxonomy, Product identification, and interaction patterns. In the target state, deeper alignment or standardisation may be required in areas such as data exchange, governance, and cross-system integration.

Overall, this analysis demonstrates that interoperability in Product Catalog implementations is achievable without enforcing a single conceptual model. Instead, it relies on a shared understanding of core building blocks, consistent management of key dependencies, and clear interface principles. By focusing on these elements, the industry can enable diverse implementations to coexist while supporting a scalable transition toward modern, product-centric retailing.

Annex 1: Participating IT Providers

accelya

Hitit

amadeus

PROS

atpco

sabre

Datalex

TURKISH TECHNOLOGY

FLYR

tpc

IATA Coordinator: Anca Dolocan, Senior Manager, Airline Retailing