EPCIS and CBV Implementation Guideline

Using EPCIS & CBV to increase supply chain visibility

Release 2.0., Ratified, Mar 2023

3 Document Summary

Document Item	Current Value
Document Name	EPCIS and CBV Implementation Guideline
Document Date	Mar 2023
Document Version	2.0
Document Issue	
Document Status	Ratified
Document Description	Using EPCIS & CBV to increase supply chain visibility

4 Roster of EPCIS/CBV 2.0 MSWG participants

First Name	Last Name	Company
Neil	Aeschliman	GS1 US
Vladimir	Alexiev	Sirma AI (Ontotext)
Mattia	Alfieri	Apio
Michael	Allen	AIM Global
Philip	Allgaier	bpcompass GmbH
Koji	Asano	GS1 Japan
Aravinda	Baliga B	benelog GmbH & Co. KG
Melissa	Banning	Gilead Sciences
Jonathan	Barnes	FoodLogiQ
Nathan	Barry	IBM (US)
Sebastian	Bartkowiak	Institute of Logistics and Warehousing
Nicolas	Becker	European EPC Competence Center GmbH (EECC)
Beth	Bernick	TraceLink
Jayson	Berryhill	Envisible LLC
Roberto	Bisignano	Xelion Tech s.r.l.
Mats	Bjorkqvist	GS1 Sweden
Sven	Böckelmann	benelog GmbH & Co. KG
Zsolt	Bocsi	GS1 Hungary
Maik	Bollmacher	T-Systems International GmbH
Cyrille	BORDIER	Axway
Klaudiusz	Borowiak	GS1 Poland
Steffen	Butschbacher	bpcompass GmbH
Jaewook	Byun	Auto-ID Labs at KAIST
Jose Manuel	Cantera Fonseca	IOTA Foundation
Kevin	Capatch	Geisinger Health System (GHS)
Karolin	Catela	GS1 Sweden
Robert	Celeste	Center for Supply Chain Studies
Patrick	Chanez	INEXTO SA
Devang	Chauhan	Brevitaz Systems
Alessandro	Chelli	Apio

Flavia	Costa	GS1 Brasil
Luiz	Costa	GS1 Brasil
Jay	Crowley	US Data Management, LLC (USDM)
Bence	Csordás	GS1 Hungary
Beth	Davis	FoodLogiQ
Kevin	Dean	GS1 Canada
Yi	Ding	GS1 China
Jeanne	Duckett	Avery Dennison RFID
Judit	Egri	GS1 Hungary
Fadi	El-Turk	GS1 UK
Jürgen	Engelhardt	Robert Bosch GmbH
Oliver	Erlenkämper	Movilizer GmbH
Emanuele	Ferro	Euranet
Fabrizio	Figueroa	GS1 Canada
Alexis	Flores	GS1 Mexico
Fabio	Forno	Xelion Tech s.r.l.
Jesper Kervin	Franke	GS1 Denmark
Achim	Fricker	QINUM GmbH
Sophie	Fuller	GS1 UK
Jason	Geyen	Optel Group
Jean-Christophe	Gilbert	GS1 France
Matt	Glassman	rfXcel Corporation
Nicole	Golestani	GS1 Canada
Heinz	Graf	GS1 Switzerland
Richard	Graves	Phy
Nadi (Scott)	Gray	GS1 Global Office
Dominique	Guinard	Digimarc
Danny	Haak	Nedap
Rami	Habbal	GS1 UAE
Dominik	Halbeisen	SBB AG
Rosemary	Hampton	Johnson & Johnson
Eileen	Harpell	GS1 Global Office
David	Harper	Delivr Corporation
Mark	Harrison	Milecastle Media Limited
Gary	Hartley	GS1 New Zealand
Philip	Heggelund	DuckScape Inc
Martin	Herold	T-Systems International GmbH
David	Hintzen	GS1 Germany
Sandra	Hohenecker	GS1 Germany
Rémy	Höhener	SBB AG
Yoshihiko	Iwasaki	GS1 Japan
Yuan	Ji	CAICT
Jia	Jianhua	GS1 China
Gokul	Kandiraju	IBM (US)
Iliada	Karali	GS1 Association Greece

Roula	Karam	Antares Vision
Steven	Keddie	GS1 Global Office
Kazuna	Kimura	GS1 Japan
Catherine	Koetz	GS1 Australia
Akshay	Koshti	Robert Bosch GmbH
Ben	Kothari	Ampliflex inc
Gergely	Köves	TE-FOOD International GmbH
Rajendra	Kulkarni	Johnson & Johnson
Chris	Lai	GS1 Hong Kong, China
Wai Shing	Lai	GS1 Hong Kong, China
Iker	Larizgoitia	Digimarc
Endre	Lazar	Movilizer GmbH
Petri	Leppänen	GS1 Finland
Zhimin	Li	GS1 China
Pedro	Lima	GS1 Portugal
Maheshwar	Lingichetty	Bracket Global
Joseph	Lipari	Systech International
Sean	Lockhead	Innovit Inc.
wayne	Luk	GS1 Hong Kong, China
Yan	Luo	GS1 China
Rob	Magee	Vantage Consulting Group
Noriyuki	Mama	GS1 Japan
Timothy	Marsh	GS1 Global Office
Tobias	Matt	Testo SE & Co. KGaA
Julie	McGill	FoodLogiQ
Jan	Merckx	GS1 Netherlands
Jochen	Metschke	Ceratizit
Andrew	Meyer	GS1 US
Doug	Migliori	ControlBEAM Digital Automation / ADC Technologies Group
Nguyen Dac	Minh	GS1 Vietnam
Mario	Mira	Dentsu Aegis Network
Jin	Mitsugi	Auto-ID Labs at Keio University
Adrien	Molines	GS1 France
Eric	Moore	Testo SE & Co. KGaA
Gena	Morgan	GS1 US
Alice	Mukaru	GS1 Sweden
Tobias	Müller	tabya GmbH
Michael	Natale	Pfizer
Zubair	Nazir	GS1 Canada
Giada	Necci	GS1 Italy
Stevan	Nesovic	OriginTrail
Alice	Nguyen	GS1 Vietnam
Falk	Nieder	European EPC Competence Center GmbH (EECC)
Masatoshi	Nomachi	Japan Pallet Rental Corporation

Jussi	Numminen	wirepas
Scott	Olson	Zebra Technologies Corporation
Onur	Önder	BLG CONTRACT LOGISTICS GmbH & Co. KG
Annie	Opel	Envisible LLC
Ciaran	O'Reilly	GS1 Ireland
Luis	Paniagua	GS1 Costa Rica
Sergio	Pastrana	GS1 Mexico
Nicolas	Pauvre	GS1 France
James	Perng	GS1 Chinese Taipei
Scott	Pugh	Jennason LLC
Branimir	Rakic	OriginTrail
Murli	Ram	PLM TrustLink
Mark	Read	Coriel Ltd
Jan	Reichert	SAP SE
Craig Alan	Repec	GS1 Global Office
Ennio	Riccobene	Xelion Tech s.r.l.
Octavio	Rodriguez	Systech International
Greg	Rowe	GS1 Global Office
Sylvia	Rubio Alegren	ICA Sverige AB
Thomas	Rumbach	SAP SE
Zbigniew	Rusinek	GS1 Poland
Bonnie	Ryan	GS1 Australia
Chuck	Sailer	CGS Consulting
Yuki	Sato	GS1 Japan
Trond	Saure	Bouvet Norge AS
Hans Peter	Scheidt	C & A SCS
Sue	Schmid	GS1 Australia
Sebastian	Schmittner	European EPC Competence Center GmbH (EECC)
Eugen	Sehorz	GS1 Austria
Nikolaos	Servos	Robert Bosch GmbH
Bhavesh	Shah	Brevitaz Systems
Marcel	Sieira	GS1 Australia
Shalika	Singh	Brevitaz Systems
Joakim	Skimutis	Foodchain
Jim	Springer	EM Microelectronic
Graham	Stanley	rfXcel Corporation
Jennie	Stitzinger	GS1 US
Christa	Suc	GS1 UK
Harald	Sundmaeker	ATB Institut für angewandte Systemtechnik Bremen GmbH
Veljko	Terzic	OriginTrail
Claude	Tetelin	GS1 Global Office
Yalew	Tolcha	Auto-ID Labs at KAIST
Elena	Tomanovich	GS1 Global Office
Viet	Tran	GS1 Vietnam

Ralph	Troeger	GS1 Germany
Alec	Tubridy	GS1 Ireland
Roman	Vaculin	IBM (US)
Bharat Reddy	Vaka	Vaka Consulting Inc
Michiel	Valee	Dockflow
Tjibbe	van der Laan	Nedap
Jeroen	van Weperen	GS1 Australia
Krisztina	Vatai	GS1 Hungary
Martijn	Veerman	Customer Value
Linda	Vezzani	GS1 Italy
Judit	Vicsai	GS1 Hungary
Vincenzo	Viola	Xelion Tech s.r.l.
Sten	Walde	GS1 Sweden
Elizabeth	Waldorf	TraceLink
John	Walker	Semaku
Amber	Walls	GS1 US
Junyu	Wang	Auto-ID Labs at Fudan University
Yi	Wang	GS1 China
Chunguang	Wang	GS1 China
Ethan	Ward	GS1 Australia
Laura	Weingarten	BLG CONTRACT LOGISTICS GmbH & Co. KG
Roman	Winter	GS1 Germany
Zhang	Wm	GS1 China
Connie	Wong	GS1 Canada
XinMin	WU	GS1 China
Ruoyun	Yan	GS1 China
Shi	Yu	Beijing REN JU ZHI HUI Technology Co. Ltd.

Log of Changes

Release	Date of Change	Changed By	Summary of Change
1.0	Oct 2015	Ken Traub	First release
1.2	Feb 2017	Ken Traub	Update to include EPCIS 1.2 features
2.0	Mar 2023	Craig Alan Repec	WR 22-270 Update to include features new to EPCIS/CBV 2.0

Disclaimer

GS1®, under its IP Policy, seeks to avoid uncertainty regarding intellectual property claims by requiring the participants in the Work Group that developed this **EPCIS and CBV Implementation Guideline** to agree to grant to GS1 members a royalty-free licence or a RAND licence to Necessary Claims, as that term is defined in the GS1 IP Policy. Furthermore, attention is drawn to the possibility that an implementation of one or more features of this Specification may be the subject of a patent or other intellectual property right that does not involve a Necessary Claim. Any such patent or other intellectual property right is not subject to the licensing obligations of GS1. Moreover, the agreement to grant licences provided under the GS1 IP Policy does not include IP rights and any claims of third parties who were not participants in the Work Group.

Accordingly, GS1 recommends that any organisation developing an implementation designed to be in conformance with this Specification should determine whether there are any patents that may encompass a specific implementation that the organisation is developing in compliance with the Specification and whether a licence under a patent or other intellectual property right is needed. Such a determination of a need for licensing should be made in view of the details of the specific system designed by the organisation in consultation with their own patent counsel.

THIS DOCUMENT IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR PARTICULAR PURPOSE, OR ANY WARRANTY OTHER WISE ARISING OUT OF THIS DOCUMENT. GS1 disclaims all liability for any damages arising from use or misuse of this document, whether special, indirect, consequential, or compensatory damages, and including liability for infringement of any intellectual property rights, relating to use of information in or reliance upon this document.

GS1 retains the right to make changes to this document at any time, without notice. GS1 makes no warranty for the use of this document and assumes no responsibility for any errors which may appear in the document, nor does it make a commitment to update the information contained herein.

Table of Contents

1 Introduction 13
1.1 Intended audience 13
1.2 Document scope 13

2 Overview of EPCIS 13
2.1 What's in the EPCIS and CBV standards? 14
2.2 Example of EPCIS Visibility Data 14
2.3 EPCIS in business applications 16
2.4 Benefits and business opportunities 17
2.5 EPCIS Data in relation to other types of data 17
2.6 How EPCIS fits into a typical IT landscape 19
2.7 EPCIS and GS1 standards 20

3 Anatomy of an EPCIS event 20
3.1 EPCIS dimensions: What, When, Where, Why, How 21
3.2 The What dimension 21
3.3 The When dimension 21
3.4 The Where dimension 22
3.5 The Why dimension 22
3.6 The How dimension 23
3.7 EPCIS Event types and action 24
3.8 EPCIS and the Core Business Vocabulary (CBV) 25
3.9 Putting it together 26

4 Designing a Visibility system using EPCIS 27
4.1 Step 1: Collect Visibility goals and requirements 27
4.2 Step 2: Document the Business Process flow 28
4.3 Step 3: Break each process flow into a series of discrete business steps 28
4.4 Step 4: Decide which business steps require visibility events 30
4.5 Step 5: Model the completion of each step as a visibility event 31
4.6 Step 6: Decide what data fields are to be included in the visibility event 33
4.6.1 Designing the What Dimension 33
4.6.2 Designing the When Dimension 34
4.6.3 Designing the Where Dimension 35
4.6.4 Designing the Why Dimension 36
4.6.5 Example 39
4.7 Step 7: Determine the Vocabularies that populate each Data Field 40
4.7.1 Vocabularies for the What dimension 40
4.7.2 Vocabularies for the Where dimension 41
4.7.3 Vocabularies for the Why dimension 42
4.7.4 Example 43
4.8 Step 8: Document the Visibility Events in a Visibility Data Matrix 44

5 Advanced EPCIS Modelling 45
5.1 Aggregation/Disaggregation 45
5.1.1 Aggregation and Disaggregation 46
5.1.2 Multiple Levels of Aggregation 47

5.2 Drop Shipment 47
5.3 Class-Level Tracing 48
5.3.1 Inherent Limitations of Traceability Using Class-Level Identification 48
5.3.2 Beginning-of-Life Events for Class-Level Identification 49
5.3.3 Class-Level Identification In Aggregation 50
5.3.4 Mixing Instance-Level and Class-Level Identification in the Same Event 50
5.4 Instance/Lot Master Data (ILMD) 51
5.5 Transformation 54
5.5.1 Transformation Event Example 55
5.5.2 Long-Running Transformations 56
5.6 Coupons and Vouchers 57
5.6.1 Simple Coupon Process 57
5.6.2 Coupon Example With Coupon Broker 58
5.7 Returnable Asset Management Using GRAI 58
5.8 User Extensions 60
5.8.1 User extensions in XML 60
5.8.2 User extensions in JSON-LD 62
5.9 Erroneous events 63
5.9.1 Example 1: Correction using an ordinary event – simple addition 64
5.9.2 Example 2: Correction using an ordinary event – corrective business step 64
5.9.3 Example 3: Declaring a prior event to be in error, with no corrective event 65
5.9.4 Example 4: Declaring a prior event to be in error, with a corrective event 66
5.9.5 Timing of capturing error declaration and corrective events 67
5.9.6 Querying for events in the presence of errors and corrections 67
5.10 Association Events 68
5.10.1 Example 1: Installing components/assemblies into larger items 68
5.10.2 Example 2: Installing components/assemblies into physical locations 68
5.11 Sensor-based quality data 69
5.11.1 Sensor example 1: Control/prove temperature compliance 69
5.11.2 Sensor example 2: Exception notification 71
5.11.3 Sensor example 3: Condition monitoring and tracking of intermodal transports 73
5.11.4 Sensor example 4: Condition monitoring and tracking of intermodal transports 79
5.12 End-of-Life Events for Instance-Level Identification 80
5.13 Inferred completeness 80

6 Sharing EPCIS Data 81
6.1 EPCIS Queries 81
6.2 Query Modes: Pull vs Push 83
6.3 The EPCIS Query Control Interface 84
6.4 Choreography Models: Sharing Data across a Supply Chain 85
6.4.1 Centralised Choreography 86
6.4.2 Distributed Push Choreography 87
6.4.3 Distributed Query Choreography 88
6.5 Synchronisation of Master Data 89
6.6 Retrieval of master data using a Web request for the GS1 Digital Link URI 90
6.7 Redaction of EPCIS Event Data 90

7 Data Validation and System Interoperability 91
7.1 Validation of EPCIS events 91

7.2 Certification program 91
7.3 Requirements of program certification 91
7.4 Non-normative tools 91

8 References 91

9 Appendix: XML and JSON-LD Examples 93

List of Figures

Figure 2-1 Simple Business Process Showing Generation of EPCIS Data 15
Figure 2-2 Overlap Between Transaction Data and Visibility Data 18
Figure 4-1 Example Business Process Flow 28
Figure 4-2 Forward Logistics Process Flow, Diagram 1 of 2 29
Figure 4-3 Forward Logistics Process Flow, Diagram 1 of 2 29
Figure 4-4 Forward Logistics Process Flow with Visibility Capture Indicated.............................31
Figure 6-1 Centralised Choreography 86
Figure 6-2 Distributed Push Choreography 87
Figure 6-3 Distributed Query Choreography...88

List of Tables

Table 2-1 Example Business Applications and Their Use of EPCIS Data 16
Table 2-2 Categories of Data in the "Share" Layer of GS1 standards 18
Table 3-1 EPCIS Event Information Content for Step V3 of Example Business Process 26
Table 4-1 Assignment of Event Types to Business Process Steps in Example Business Process ...32
Table 4-2 Class and Instance Level Object Identification 33
Table 4-3 Source/Destination Types Defined in Core Business Vocabulary 39
Table 4-4 EPCIS Event Information Content for Step V4 of Example From Section 4.4 40
Table 4-5 Examples of Standard Vocabulary Identifiers Defined in Core Business Vocabulary 42
Table 4-6 Example Assignment of Identifiers for EPCIS Event From Section 4.6 43
Table 4-7 Example Visibility Data Matrix 44
Table 5-1 Examples of Commonly Occurring Aggregations 45
Table 5-2 Action Values for Aggregation Events 46
Table 5-3 Example EPCIS Aggregation Event Information Content 46
Table 5-4 Example EPCIS Aggregation Event Information Content for a Two-Level Hierarchy 47
Table 5-5 Source/Destination Types Defined in the Core Business Vocabulary 47
Table 5-6 EPCIS Event Information Content for Example "Drop Shipment" Scenario 48
Table 5-7 Example EPCIS Event Information Content Using Class-Level Identification. 49
Table 5-8 EPCIS Event Information Content for Aggregation of Children Identified at Class Level 50
Table 5-10 EPCIS Aggregation Event Information Content with Children Identified at Both Instance and Class Level 51
Table 5-12 EPCIS Event Information Content Showing Instance/Lot Master Data (ILMD) 52
Table 5-13 Examples of Transformation Business Processes 55
Table 5-14 Example EPCIS Transformation Event Information Content 55
Table 5-15 Example EPCIS Transformation Event Information Content Linked Via Transformation ID 56
Table 5-16 Example EPCIS Event Information Content for Simple Digital Coupon Business Process 57
Table 5-17 Process Flowchart for Example Returnable Asset Management Business Process 59
Table 5-18 EPCIS Event Information Content for Returnable Asset Management Business Process (V1 – V4) 59
Table 5-19 EPCIS Event Information Content for Returnable Asset Management Business Process (V5 – V9) 59
Table 5-20 EPCIS Event information content illustrating user/vendor extensions 60
Table 5-21 Example of correcting an error by adding an ordinary event with a corrective business step. 64
Table 5-22 Example of correcting an error by adding an ordinary event. 65
Table 5-23 Example of correcting an error by adding an error declaration event. 65
Table 5-24 Example of correcting an error by adding an error declaration event. 66
Table 6-1 Selected EPCIS Query Criteria 81
Table 6-2 Examples of Business Information Needs and Corresponding EPCIS Query Criteria 82
Table 6-3 Example Business Scenarios and Corresponding Likely EPCIS Query Modes 84
Table 6-4 EPCIS Query Control Interface Operations 84
Table 6-5 Characteristics of Centralised Choreography 86
Table 6-6 Characteristics of Distributed Push Choreography 87
Table 6-7 Characteristics of Distributed Query Choreography 88

1 Introduction

Consumers and businesses rely on global supply chains to produce a diverse array of high quality, safe goods and services at affordable prices in a socially and environmentally responsible way. Meeting the demands of today’s consumer requires a much finer degree of supply chain visibility than has been typically exposed in the past. Increasingly, organisations, governments and consumers want that ability to track and trace the products they purchase, the things they eat and perhaps even electronic records about things they care about.

Visibility data can describe the origin of an object (virtual or physical), each location where it is subject to a business process throughout the supply chain or other process, when those processes took place and what was occurring to that object at each point. Visibility data is the WHAT, WHERE, WHEN, WHY and HOW about an object. Capturing and sharing visibility data, either internally or across trading partners provides a view into the history of the manufacture, shipping, receiving and selling processes that allow for a more efficient, affordable and safe supply chain.

EPCIS is a GS1 standard that defines a common data model for visibility data and interfaces for capturing and sharing visibility data within an enterprise and across an open supply chain. The goal of EPCIS is to enable disparate applications to create and share visibility event data, both within and across enterprises. Ultimately, this sharing is aimed at enabling users to gain a shared view of physical or digital objects within a relevant business context.

1.1 Intended audience

This guide is intended to provide supply chain stakeholders, including manufacturers, distributors, retailers, logistics providers, solution providers, business process architects, IT departments (developers) and solution providers with an introduction to implementing a visibility system using EPCIS and the Core Business Vocabulary (CBV) specifically, along with other GS1 standards.

1.2 Document scope

This guide was developed to provide both overview and guidance on getting started with visibility systems using EPCIS. It is not intended to be a detailed, technical industry-specific "how to" guide. Industries including Pharmaceutical, Electronics, Logistics and Food & Agriculture, have developed industry specific implementation guides for EPCIS. This document intends to provide guidance at a basic use or foundational level, allowing those guidelines to layer on their specific industry requirements on top.

2 Overview of EPCIS

The goal of EPCIS is to enable disparate applications to create and share visibility event data, both within and across enterprises. Ultimately, this sharing is aimed at enabling users to gain a shared view of physical or digital objects within a relevant business context.

"Objects" in the context of EPCIS typically refers to physical objects that are identified either at a class or instance level and which are handled in physical handling steps of an overall business process involving one or more organisations. Examples of such physical objects include trade items (products), logistic units, returnable assets, fixed assets, physical documents, etc. "Objects" may also refer to digital objects, also identified at either a class or instance level, which participate in comparable business process steps. Examples of such digital objects include digital trade items (music downloads, electronic books, etc.), digital documents (electronic coupons, etc.), and so forth. Throughout this document the word "object" is used to denote a physical or digital object, identified at a class or instance level, that is the subject of a business process step. EPCIS data consist of "visibility events," each of which is the record of the completion of a specific business process step acting upon one or more objects.

The EPCIS standard was originally conceived as part of a broader effort to enhance collaboration between trading partners by sharing of detailed information about physical or digital objects. The name EPCIS reflects the origins of this effort in the development of the Electronic Product Code (EPC). It should be noted, however, that EPCIS does not require the use of Electronic Product Codes, and does not even require instance-level identification. EPCIS/CBV 2.0 permits the use of a constrained set of GS1 Digital Link URIs as an equivalent to existing EPC URNs or generic HTTP URLs.

The EPCIS standard applies to all situations in which visibility event data is to be captured and shared, and the presence of "EPC" within the name is of historical significance only.

2.1 What’s in the EPCIS and CBV standards?

The EPCIS standard defines:

- A **data model** for visibility event data, with XML (Extensible Markup Language), JSON (JavaScript Object Notation) and JSON-LD (JavaScript Object Notation for Linked Data) syntax support.
- Open, standardised **interfaces** that allow for seamless integration of well-defined services in inter-company environments as well as within companies. There are two interfaces defined in the EPCIS standard:
    - A **capture interface** through which visibility event data conforming to the EPCIS data model may be delivered from capturing applications to a receiver, typically a persistent repository of EPCIS data; and
    - A **query interface** through which EPCIS event data may be requested by and delivered to a business application or a trading partner.

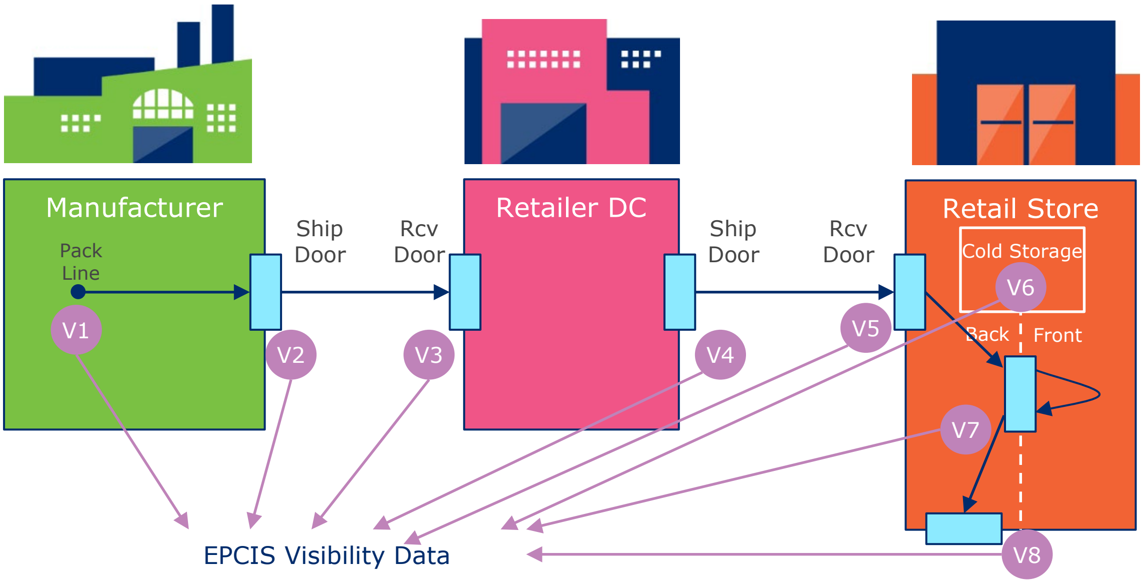
Standard interfaces are defined in the EPCIS standard to enable visibility event data to be captured and queried using a defined set of service operations and associated data standards, all combined with appropriate security mechanisms that satisfy the needs of user companies. In many or most cases, this will involve the use of one or more persistent databases of visibility event data, though a direct linkage between capture and query interface could be used for direct application-to-application sharing without persistent databases.

EPCIS is intended to be used in conjunction with the CBV [CBV2.0]. The CBV provides definitions of data values that may be used to populate the data structures defined in the EPCIS standard. The use of the standardised vocabulary provided by the CBV standard is critical to interoperability and critical to provide for querying of data by reducing the variation in how different businesses express common intent. Therefore, capturing applications should use the CBV standard to the greatest extent possible in constructing EPCIS data.

2.2 Example of EPCIS Visibility Data

EPCIS data is intended to provide information systems with visibility as to where objects are (and have been) within the business processes in which those things are handled. The following figure illustrates a simple business process, showing where EPCIS data may be generated.

Figure 2-1 Simple Business Process Showing Generation of EPCIS Data



This figure illustrates a simple business process in which a trade item is manufactured and shipped to a distribution centre, where it is subsequently received and later shipped to a retail store, where it is received and later moved into the sales area. The entire business process may be viewed as a sequence of individual business steps: product packaging, packing into a shipping container, shipping, receiving, and so on. EPCIS data can provide a detailed record of any or all of these steps. A unit of EPCIS data that describes the completion of one business step is called an EPCIS *event*, and a collection of EPCIS events provides a detailed picture of a business process over time and place.

For example, a single EPCIS event records the receipt of one shipment at the distribution centre. The information content of this event is organised into five dimensions:

- ***What:*** Information about what trade items and/or shipping containers were received
- ***When:*** The date and time when receiving occurred, and the local time zone in effect
- ***Where:*** The location where the shipment was received, and where the items are expected to be following the event
- ***Why:*** Information about the business context, including:
  - indication that the business step is a receiving operation (as opposed to shipping or some other business step);
  - information on the asset's status (e.g., that the shipment is in transit)t;
  - the identity of the shipping and receiving locations, as well as the identity of the source and destination parties that are involved in possession or ownership;
  - Links to relevant business transaction documents, such as a purchase order, an invoice, a despatch advice (a.k.a. advance ship notice), etc.
- ***How:*** Sensor-based conditional information, captured – for example, during refrigerated transport, or in the retailer's cold storage room – either in predefined time intervals or when a specific temperature threshold is exceeded.

Each of the business steps in the process illustrated in the figure above could be the source of an EPCIS event. The details of the content of each of those events are different depending on the business step, but all have the same four- or five-dimensional structure.

2.3 EPCIS in business applications

The power of EPCIS lies in bringing together individual events that are recorded over time and across a complete business process and/or supply chain. Examples of such paradigms include:

- Finding the most recent EPCIS event for a given object, to learn where it currently is and what state it is in ("tracking");
- Assembling a history of events for a given object, to understand its path through an overall business process or supply chain ("tracing");
- Analysing a collection of events gathered over time at a particular location or within a particular business process; ("analysis")
- Comparing the actual status of objects based on a current EPCIS event to what was expected to have happened based on a prior business transaction or a prior EPCIS event; ("checking")
- Triggering other business processes in real time based on what a freshly captured EPCIS event reveals about the completion of a business step ("automation").

Below are examples of business applications that can benefit from EPCIS data, along with the paradigm involved. It should be noted, however, that these paradigms are broad generalisations, and in reality a business application may make use of EPCIS data in a variety of ways that combine or step outside paradigms.

Table 2-1 Example Business Applications and Their Use of EPCIS Data

Business Application	How EPCIS Data Is Used	Primary Paradigm
Anti-counterfeiting, Provenance	Validate origin and pedigree of product	Tracing, Checking
Chain of custody/ownership	Document and reproduce product attributes and all partners that had physical possession of a product	Tracing
Couponing	Customer behaviour analysis and real-time coupon validation	Analysis, Checking
Customs clearance	Improve customs efficiency, reduce fraud with electronic seals	Tracing
Recall	Speed recalls due to precise traceability of products of concern	Tracking (to find recalled product), Tracing (to monitor progress of recall)
Sales promotion	Ensure that promotional goods reach consumers at the right place and time	Tracking
Traceability	Trace product movement forward and backward through specified stages of the extended supply chain.	Tracing
Business Process Optimisation	Shorten lead times, increase capacity utilisation, improve delivery quality and accuracy	Automation, Analysis
Exception Management	Alert process owners of deviation from desired product, timing, quantity, quality, location, status	Checking, Automation
Food Freshness	Monitoring whether expiration dates are not exceeded	Tracking, Automation
Asset Management	Keeping track of fixed assets and ensuring that adequate quantities are available to the business processes that need them	Tracking, Analysis
Inventory Management	Capture inventory inputs, outputs, stock taking	Tracking, Analysis
Process Documentation	Automate digital document generation and workflow, link to documents, products and locations identified with GS1 keys	Automation

Each one of these applications could be deployed in one of three modes:

- Internal: The business process exists within the facilities and is under the control of a single organisation.
- External, Closed Chain: The business process spans more than one organisation, but all organisations involved are known in advance.

- External, Open Chain: The business process spans more than one organisation, and the set of organisations involved is not known in advance and changes over time. This mode is typical of large supply chains involving mutual trade.

In all three modes, a key element of solution design is to determine the proper data content of EPCIS events so that the requirements of the business applications are met. In the external modes, an additional consideration is the design of the way that EPCIS events are communicated between the multiple organisations involved (often referred to as the "choreography" in contrast to the "content").

In the external, open chain mode, the value of EPCIS and CBV being open standards is obvious: when all parties adhere to a standard, it is possible to achieve interoperability and mutual understanding of data even without prior collaboration of the parties on solution design. However, this is just as important in a closed chain or even a strictly internal application—primarily because internal applications tend to become external and closed applications tend to be come open over time. It is therefore important to follow best practices for external, open applications even when designing a closed or purely internal application.

2.4 Benefits and business opportunities

Enhanced visibility offers a number of various benefits at all points in the supply chain in all industries. A record of processes at the point of origin or manufacture, through various distribution points, to the final point of sale to a consumer offers the potential for benefits including:

- Optimised receiving productivity
- Improved inventory management
- Increased pick rates
- Reduced errors in mispicks and shorts
- Improved order accuracy and reduces billing errors
- Better product and location identification throughout track and trace processes
- Increased operational efficiencies across various business processes
- Improved preparedness for fast and precise recalls
- Enhanced consumer protection

EPCIS and its companion standard, the CBV, provide a technical foundation for capturing and sharing visibility data. It helps answer the questions "where is something and where has something been?" Sharing visibility data in a standard manner offers significant advantages over proprietary solutions. EPCIS allows for sharing of data between various business applications, either internally or between trading partners. EPCIS facilitates real time processing and return of event based data, both streaming (inflow and outflow of events) and complex event processing (match filtering of events). An EPCIS based system supports the demands of the consumer’s growing appetite of more and more product information, including the path the things they are purchasing have travelled.

It is important to note that EPCIS is a set of interface standards, one for capturing the data and one for querying the data. The CBV provides the business context to the data model prescribed in EPCIS. Many software applications focused on traceability or other business processes that may benefit from visibility data within and across organisation implement EPCIS as a foundation. Indeed, organisations looking to develop a visibility strategy should look for solutions based on this standard.

2.5 EPCIS Data in relation to other types of data

GS1 standards in the "Share" layer pertain to three categories of data that are shared between end users:

Table 2-2 Categories of Data in the "Share" Layer of GS1 standards

Data	Description	GS1 Standards
Master Data	Data, shared by one trading partner to many trading partners, that provides descriptive attributes of real-world entities identified by GS1 Identification Keys, including trade items, parties, and physical locations.	GDSN Online retrieval via GS1 Digital Link, leveraging GS1 Web Vocabulary
Transaction Data	Trade transactions triggering or confirming the execution of a function within a business process as defined by an explicit business agreement (e.g., a supply contract) or an implicit one (e.g., customs processing), from the start of the business process (e.g., ordering the product) to the end of it (e.g., final settlement), also making use of GS1 Identification Keys.	EANCOM, GS1 XML
Visibility Data	Details about physical or digital activity in the supply chain of products and other assets, identified by keys, detailing where these objects are in time, and why; not just within one organisation’s four walls, but across organisations.	EPCIS

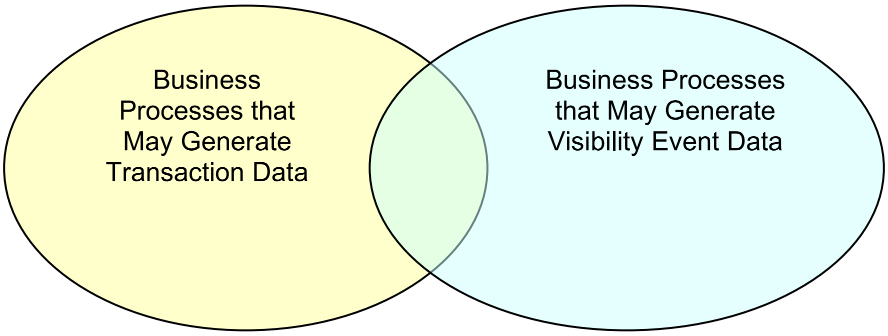
As the table suggests, visibility data (EPCIS event data) is a *new* type of data, different in character from either master data or transaction data.

A chief distinguishing characteristic of EPCIS data is that it occurs in much greater volume than either master data or transaction data. Like transaction data (and unlike master data), new visibility data is generated continuously as an organisation conducts more business. But visibility data occurs in greater volume because:

- Visibility data frequently refers to individual instances of objects, for example trade items identified by the combination of a Global Trade Item Number (GTIN) and a serial number.
- Even when visibility data refers to objects at the class level, visibility data is generated at more steps within an overall business process. For example, a trade item flowing from manufacturer to retailer may be subject to just a single business transaction (the sale from manufacturer to retailer) but be the subject of several dozen visibility events as it progresses through the manufacturer’s and retailer’s facilities.
- Visibility data often has historical value for traceability, and so may be retained for longer periods of time than business transaction data.

Visibility data is complementary to transaction data, as some visibility events occur in the absence of business transactions and conversely some business transactions take place without handling of objects. Where the same business process simultaneously yields visibility data and transaction data, they provide complementary data.

Figure 2-2 Overlap Between Transaction Data and Visibility Data


Business Processes that May Generate Transaction Data
Business Processes that May Generate Visibility Event Data

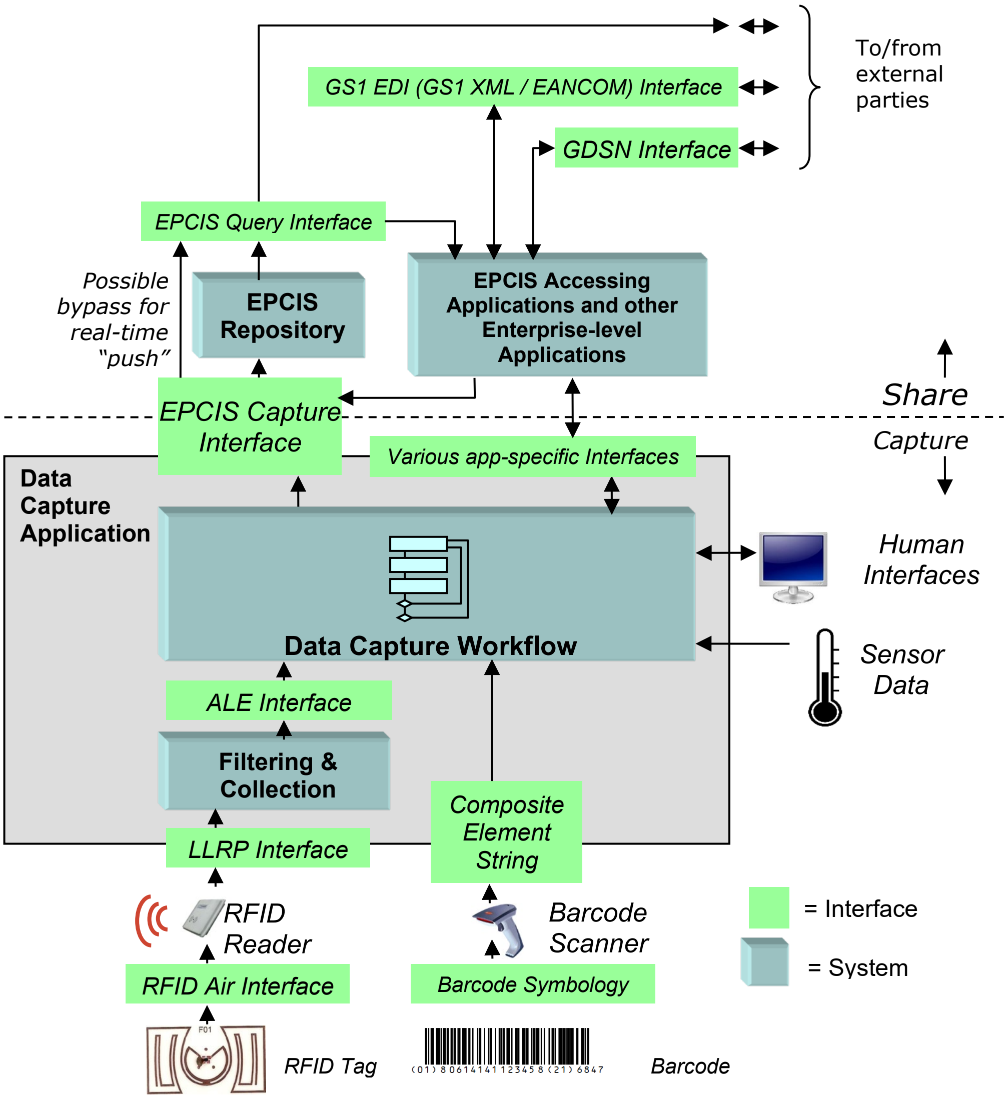
Examples of all three possibilities:

- In some cases, a visibility event coincides with a business transaction, so that there may be a piece of transaction data and a piece of visibility event data describing different aspects of the same occurrence. For example, when goods are shipped from a loading dock, there may be a despatch advice (a piece of transaction data that confirms the sender’s intent to deliver specific goods to the receiver) and an EPCIS event with business step "shipping" (a piece of visibility data that confirms the observation of goods leaving the loading dock). Even in such cases, the transaction data and visibility event data may not be in 1:1 correspondence; for example, a single despatch advice may correspond to several visibility events if different parts of the shipment are handled separately.

- A visibility event may occur with no corresponding business transaction. For example, when a trade item moves from the "back room" storage of a retail store to the sales area where a consumer can purchase it. This is a highly relevant event for purposes of assessing availability of product to consumers but it has no associated business transaction.
- A business transaction may take place with no corresponding visibility event. For example, when a purchaser sends an "order" message to a supplier, there is a legal interaction, but nothing occurring in the physical world where the ordered products reside (in fact, the ordered products may not even exist when the order is sent).

2.6 How EPCIS fits into a typical IT landscape

The following simplified diagram shows how EPCIS fits in to a typical company IT infrastructure.



For the sake of discussion, this picture lumps together as "back-end applications" all of the IT components that process master data and transaction data (as defined in the previous section). The specific legacy components in use will, of course, vary widely from company to company; typical components include Enterprise Resource Planning (ERP) systems, Warehouse Management Systems (WMS), Master Data Management (MDM) systems, etc.

Because visibility data is a new type of data, and as discussed in the previous section visibility data often occurs in far greater quantities, it is common that new IT components are dedicated to the processing of visibility data. These components include:

- **EPCIS Repository**: A persistent store for visibility data, including all EPCIS events generated internally within the organisation and whatever EPCIS events are received from trading partners.
- **EPCIS Capture Applications**: Software applications deployed at the "edge" of an enterprise—in factories, warehouses, stores, etc—that generate EPCIS events as business process steps are completed.
- **EPCIS Accessing Applications**: Software applications at the enterprise level that process EPCIS events to meet enterprise objectives (e.g., the objectives described in section 2.3). An EPCIS accessing application might be a simple connector to a back-end application, or a complex application that carries out some new business task using EPCIS data.

The EPCIS standard defines two interfaces:

- The **EPCIS Capture Interface**, by which the EPCIS Capture Applications deliver EPCIS events to an EPCIS Repository (or possibly directly to an EPCIS Accessing Application, in case of real-time processing)
- The **EPCIS Query Interface**, by which EPCIS Accessing Applications retrieve previously stored EPCIS event data.

In addition, the following interactions between IT components are typical:

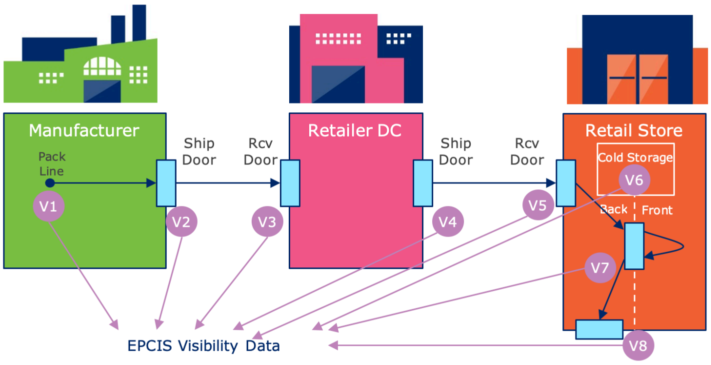
- Quite often an EPCIS Capture Application receives input from Automatic Identification and Data Capture (AIDC) devices such as bar code scanners and RFID readers (including associated RFID filtering and collection software), especially when the reading of a bar code or RFID tag is the trigger to recognise that a business process step has taken place.
- An EPCIS Capture Application may interface to one or more back-end applications to obtain relevant business context information, such as product master data or purchase order information about a shipment being received.
- An EPCIS Accessing Application may interface to one or more back-end applications either to obtain relevant business context information or to deliver new information derived from EPCIS event data (or both).
- An EPCIS Accessing Application may mediate the exchange of EPCIS data with trading partners.

2.7 EPCIS and GS1 standards

The GS1 system of standards includes standards to identify, capture, and share information about objects in supply chains. EPCIS fits in as one of the standards in the "share" group, complementing other GS1 data sharing standards for master data and transaction data, as described in section 2.5. The standards in the "identify" group provide the identifiers for real-world objects, allowing those objects to be referenced by EPCIS events. The standards in the "capture" group link the physical world to the world of information, and as noted in section 2.6 they often provide the inputs to EPCIS capture applications.

3 Anatomy of an EPCIS event

The information in an EPCIS event records the essentials of what happened during a step of a business process in which physical or digital objects were handled, expressed via the four dimensions of *what*, *where*, *when*, *why* and, if applicable, *how*. This section looks in detail at one EPCIS event for a specific business process step to show exactly how those four dimensions are populated. Section 4 goes on to explain how to design an EPCIS for any business process step.



The business process step illustrated in this section is Step V3 from the example process flow described in section 2.2. In the overall process, a trade item is manufactured and shipped to the distribution centre of a retailer, which subsequently ships it to a retail store. Step V3 is the step where the trade item is received from the manufacturer at the retailer's distribution centre. In this example, we will further assume that the trade item is a large consumer product such as a bicycle or a television set; this avoids having to consider complexities such as items packed into cases or cases stacked on a pallet. The shipment in this example consists of a single trade item identified by a GTIN plus serial number.

The EPCIS event for Step V3 includes the following data:

- The *What* dimension identifies the product that is received; in this case, using the GTIN and serial number of the product.
- The *When* dimension indicates when the receiving operation took place.
- The *Where* dimension says where the product was received, namely the distribution centre of the retailer
- The *Why* dimension provides the business context. This includes identifying the step of the business process as "receiving," indicating that the state of the product is that it is progressing normally through the forward supply chain, linking to business transaction documents such as the governing purchase order and invoice, and identifying the parties to the transfer of ownership (i.e., the manufacturer and the retailer).
- The *How* dimension, to accommodate sensor data, if available.

The following sections discuss the information content of these data dimensions in more detail.

3.1 EPCIS dimensions: What, When, Where, Why, How

3.2 The What dimension

The *What* dimension of an EPCIS event identifies the physical or digital objects that were involved in the event. As explained in the GS1 General Specifications and the GS1 Tag Data Standard, trade items are identified using a GTIN, a GTIN plus batch/lot number, or a GTIN plus a serial number. Pallets or logistics units are identified with an SSCC. Other GS1 object identifiers include GDTI for documents, GIAI for individual assets, GRAI for returnable assets, GSRN for services, GCN for coupons, and CPID for components or parts.

In Step V3 of the example, we have a trade item identified by a GTIN plus serial number, also known as a Serialised SGTIN (SGTIN), so the *what* dimension of the EPCIS event for Step V3 contains the SGTIN of the trade item being received.

3.3 The When dimension

The *When* dimension of an EPCIS event says when the event took place. There are three data elements that are part of this dimension:

- **Event Time**: The date and time at which the event took place.
- **Event Time Zone Offset**: The time zone in effect at the place and time of the event. This is useful when an application wants to display the event time using the local time; for example, if

a package is shipped from California to Brussels, the event time zone offset can be used to display the ship date/time in US Pacific time and the receiving date/time in Central Europe time.

- **Record Time**: The date and time when the EPCIS event was recorded into an EPCIS repository. Unlike all other fields in the EPCIS event, the record time is not filled in when the event is captured nor does it describe anything about the business step taking place during the event. Record Time is a bookkeeping mechanism that helps when querying an EPCIS repository; with the record time you can tell whether an event returned from a query is a new event since the time of your last query.

In Step V3 of the example, the Event Time is the date and time when the product was received, and the Event Time Zone Offset records the time zone in effect then and there.

3.4 The Where dimension

The *Where* dimension of an EPCIS event captures where the event physically took place and/or where things are following the event.

EPCIS events allow for two location types, `readPoint` and `businessLocation` The `readPoint` is the location where the event took place. The `businessLocation` is the location where the object(s) is now considered to reside until a subsequent event takes place. Locations may be identified using a GS1 Global Location Number (GLN), a GLN plus an extension, an industry identifier other than GLN or using geo-coordinates.

For example, a box may be scanned as it passes through a door portal. The portal it passes through may be the point in which the event is captured. Someone may be physically standing there reading it through the door, or there may be a door portal reader capturing the event. This would be the `readPoint.` After the boxes passes through the portal, it now sits in a particular location. This location where the box now sits would be the `businessLocation.` Locations can be identified at a very fine level of granularity (a specific bin in a specific spot in a warehouse), in which case a GLN plus an extension may be necessary. If a location is described at a more general level (a building), a GLN may suffice. It is important to understand how locations will be identified for the purposes of capturing visibility data.

Note, it is vitally important that the master data about locations are synchronised between internal systems or trading partners so when EPCIS refers to location using a GLN or SGLN, one can be assured that all concerned understand the location in the same way.

In Step V3 of the example, the Read Point is the location where the product was received, which for the purposes of the example we assume to be a specific loading dock door of the Retailer's D.C., identified by a GLN with extension. The Business Location is the location where the product resides after it is received, which for the purposes of the example we assume to be the Retailer's D.C. with no specific place within the D.C. identified. The Business Location is in that case identified by a GLN without an extension.

3.5 The Why dimension

The *Why* dimension of an EPCIS event describes the business context in which the event took place. It can include any combination of the following data elements:

- **Business Step**: identifies what was taking place from a business perspective at the time of the event; that is, what step of a business process was occurring. Examples include "`commissioning`", "`creating_class_instance`", "`inspecting`", "`packing`", "`picking`", "`shipping`", "`retail_selling`." The CBV Standard, discussed further in section 3.8, includes a list of standard business step values.
- **Disposition**: identifies the business condition subsequent to the event of the physical or digital objects named in the What dimension. Example dispositions include "active", "in_progress", "`in_transit`", "`expired`", "recalled", "`retail_sold`" and "`stolen`." The CBV includes a list of standard Disposition values.
- **Business Transaction List**: identifies one or more particular business transactions that are relevant to an event. A business transaction is identified by a pair of identifiers: one identifier that says what type of business transaction is referenced, and a second identifier that names

the particular business transaction of that type. Examples of business transaction types are purchase order ("po"), bill of lading ("bol"), despatch advice ("desadv"). The GS1 CBV includes a list of standard business transaction type values.

- **Source List** and **Destination List**: is used to provide additional business context when an EPCIS event is part of a business transfer of ownership, responsibility or custody. As with business transactions, a source or destination is identified by a pair of identifiers: the type of the source or destination and an identifier of the source or destination of that type. The GS1 CBV (section 7.4.2) distinguishes three standard source/destination types: "owning_party", "possessing_party", "location".

In Step V3 of the example, the following values might populate the *Why* dimension of the EPCIS event:

- **Business Step**: The business step receiving defined in the CBV.
- **Disposition**: The disposition in_progress, defined in the CBV, indicating that the product is moving normally through the forward supply chain.
- **Business Transaction List**: There might be two relevant transactions: the Retailer’s purchase order, and the Manufacturer’s invoice.
- **Source** and **Destination**: The source owning party is the Manufacturer and the destination owning party is the Retailer.

3.6 The How dimension

The How dimension of an EPCIS event - optional in its entirety - can accommodate a variety of sensor data pertaining to the EPCIS event it is part of.

The term 'sensor data' covers a huge set of conceivable contents. The developed framework allows for ample flexibility: organisations are not only able to transmit physical measurements (e.g. temperature values expressed in degrees Celsius or Kelvin), but also output values of smart sensor devices, which abstract from raw sensor data. For instance, instead of a specific weight value, a simple smart sensor device would transmit a meaningful value such as 'too heavy' or 'incomplete'. Moreover, it is also possible to capture the concentration of microorganisms (e.g. bacteria) or chemical substances. In addition, there is also a selected set of statistical measures (e.g. mean value) that can be included.

All data related to the How dimension is part of the sensorElement field. The sensorElement field has two child elements:

- □ at least one sensorReport element
- □ one optional sensorMetaData element

Each of these elements contains a set of pertinent attributes, which can be outlined as follows:

sensorMetaData fields

Context	Attribute	Meaning
time	time	Time of observation
	startTime	Earliest time of observation period
	endTime	Most recent time of observation period
source	deviceID	Device from which data originates
	deviceMetadata	Location of document specifying device meta data
	rawData	Location of raw sensor data
	dataProcessingMethod	Location of document specifying data processing method
	bizRules	Location of document specifying business rules

sensorReport fields

Context	Attribute	Meaning
time	`time`	Time of observation
source	`deviceID`	Device from which data originates
	`deviceMetadata`	Location of document specifying device meta data
	`rawData`	Location of raw sensor data
	`dataProcessingMethod`	Location of document specifying data processing method
type	`type`	Property identifier
	`microorganism`	Microorganism species identifier
	`chemicalSubstance`	Chemical substance identifier
value	`value`	Quantitative (double-precision float) value of a property
	`component`	Dimension indicator of a vector value
	`stringValue`	String value of a property
	`booleanValue`	Boolean value of a property
	`hexBinaryValue`	HexBinary value of a property
	`uriValue`	URI value of a property
	`uom`	Unit of measure of specified property values
statistics	`minValue`	Minimum quantitative value of a property
	`maxValue`	Maximum quantitative value of a property
	`meanValue`	Arithmetic mean of quantitative property values
	`sDev`	Standard deviation of quantitative property values
	`percRank`	Percentile rank
	`percValue`	Percentile value

3.7 EPCIS Event types and action

The four or five dimensions that describe what is happening to an object in the physical or virtual world are captured in one of five types of an "EPCIS Event". The following is a high level summary of EPCIS event types. For details, see section 7.4 in the EPCIS 1.1 Standard.

- **EPCISEvent**: generic base class for all event types.
  - **ObjectEvent**: represents an event that happened to one or more physical or digital objects. For example shipping or receiving a pallet using the pallet’s SSCC. This is the simplest type of event, as well as the most commonly used.
  - **AggregationEvent**: represents an event that happened to one or more objects that are physically aggregated together or disaggregated from each other. For example, aggregating cases onto a pallet, or removing cases from a pallet. This is the next most common type of event after ObjectEvent, and these two event types together will cover the vast majority of events in a typical business process.

  - **TransformationEvent**: represents an event in which input objects are fully or partially consumed and output objects are produced, such that any of the input objects may have contributed to all of the output objects. For example, consider mixing batter and chocolate chips into cookie dough, then baking the dough into a batch of cookies. Once the ingredients are "transformed", the resulting product is packaged and labelled with an EAN or UPC that represents "consumer package of chocolate chip cookies" and can be scanned at retail.
  - **TransactionEvent**: represents an event in which one or more objects become associated or disassociated with one or more identified business transactions. For example, linking the pallet and cases of chocolate chip cookies to a commercial invoice.
  - **AssociationEvent**: represents an event in which objects are associated with physical locations, especially suited to capture parent-child relationships that persist even after more temporarily linked children are disassociated from the parent. For example, linking a sensor to the container or returnable asset to which it is attached; the sensor remains attached, even if the contents it is monitoring are removed or replaced.

Each event type (except for TransformationEvent) is also further qualified by the "action"; see section 4.5 of this guideline for details.

3.8 EPCIS and the Core Business Vocabulary (CBV)

The Core Business Vocabulary (CBV) specifies various vocabulary elements and their values for use in conjunction with the EPCIS standard [EPCIS1.2], which defines mechanisms to exchange information both within and across organisation boundaries. The vocabulary identifiers and definitions are prescribed to ensure that all parties who exchange EPCIS data using the CBV will have a common understanding of the semantic meaning of that data.

This CBV is intended to provide a basic capability that meets the above goal. In particular, this standard is designed to define vocabularies that are *core* to the EPCIS abstract data model and are applicable to a broad set of business scenarios common to many industries that have a desire or requirement to share data. It intends to provide a useful set of values and definitions that can be consistently understood by each party in the supply chain.

Additional end user requirements may be addressed by augmenting the vocabulary elements within with additional vocabulary elements defined for a particular industry or a set of users or a single user.

The CBV includes identifier syntax (URI structure) and specific vocabulary element values with their definitions for these *Standard Vocabularies*:

- Business step identifiers
- Disposition identifiers
- Business transaction types
- Source/Destination types
- Error reason identifiers

The CBV provides identifier syntax options for these *User Vocabularies*:

- Objects
- Locations
- Business transactions
- Source/Destination identifiers
- Transformation identifiers
- Event identifiers

The CBV provides *Master Data Attributes and Values* for describing Physical Locations, Parties, and Trade Items, including Trade Item master data attributes at the GTIN level, lot level, and instance level.

3.9 Putting it together

Putting together the four dimensions of *What*, *Where*, *When*, *Why* and (optionally) *How* yields the complete information content of an EPCIS event. The following table summarises the information content of the EPCIS event for Step V3 as discussed above:

Table 3-1 EPCIS Event Information Content for Step V3 of Example Business Process

Dim	Data Element	Contents	Comments
	Event Type	Object Event	
	Action	OBSERVE	
What	**EPC List**	A list containing one element: *GTIN* 10614141123459 *Serial* 12345	Identifies the product that was received
When	**Event Time**	Sep 23, 2012, at 10:12am UTC	The moment in time when the product was received
	Event Time Zone Offset	−05:00	Local time is five hours earlier than UTC
Where	**Read Point**	*GLN* 5012345678900 *Extension* D123	The place where the product was received, in this case a specific loading dock door at the D.C.
	Business Location	GLN 5012345678900	The place where the product is expected to be following the event, in this case the entire D.C.
Why	**Business Step**	`receiving` (from CBV)	A standard identifier defined in CBV 1.1 to indicate this is a receiving business step
	Disposition	`in_progress` (from CBV)	A standard identifier defined in CBV 1.1 to indicate the product is moving normally through the forward supply chain
	Business Transaction List	A list containing two business transaction references: Purchase Order: *GLN* 5012345000015 *PO#* ABC123 Invoice: *GLN* 0614141000012 *Inv#* XYZ987	Each business transaction reference is qualified with a GLN to make it globally unique and to identify the system or party that generated the number. "Purchase Order" and "Invoice" are standard identifiers defined in CBV 1.1 to identify business transaction types.
	Source List	A list containing one source: owning party: *GLN* 0614141000012	Receiving is a step within an overall transfer of ownership from source to destination. Here, the owning party at the source (the shipper) is identified by its GLN. "`owning_party`" is a standard identifier defined in the CBV to identify a type of source
	Destination List	A list containing one destination: owning party: *GLN* 5012345000015	Receiving is a step within an overall transfer of ownership from source to destination. Here, the owning party at the destination (the receiver) is identified by its GLN. "`owning_party`" is a standard identifier defined in the CBV to identify a type of destination

Dim	Data Element	Contents	Comments
How	**sensorElement**	Contains an optional `sensorMetadata` element and one or several `sensorReport` elements. In this case, a `Sensor Report` element is used to express an ambient temperature of 16.5 degrees Celsius at the Read Point where `receiving` is captured.	Capture of ambient temperature while a shipment is in transit and at important way points, including but not limited to the point of receiving, can be used to monitor the cold chain for temperature-sensitive assets.

Section 4 describes the design process in more detail, showing how this eventually results in EPCIS data conforming to the standard.

4 Designing a Visibility system using EPCIS

Building visibility systems requires both technical understanding of the EPCIS standard and a structured methodology. The following methodology is used to analyse a visibility process from a business perspective regardless of the technology used to capture events. Once a process is fully mapped, visibility events are identified and described. The technical details at the device level are omitted in this guide since we are primarily concerned with the business application of EPCIS data.

The visibility modelling methodology has these steps:

1. Collect visibility goals and requirements
2. Document the business process flows
3. Break each process flow into a series of discrete business steps
4. Decide which business steps require visibility events
5. Model the completion of each step as a visibility event - Understand what information is needed from a business application’s perspective
6. Decide what data fields are to be included in the visibility event
    a. Start with standard EPCIS data fields
    b. Define extension fields if necessary
7. Determine the vocabularies that populate each data field according to section 7 and 8 of the CBV standard
8. Document the visibility events in a Visibility Data Matrix

We will illustrate these steps using a simple forward logistics example. Later sections of the document describe considerations arising in other scenarios.

4.1 Step 1: Collect Visibility goals and requirements

As more and more requirements are placed on organisations to track and trace the movement of things through the supply chain, it is important to place an emphasis on the overall goals and objectives of deploying a visibility system. "What problem are we trying to solve"?

The goal may be to meet a governmental regulation, or for improving efficiencies in the shipping process, or to ensure a high level of customer service by knowing where something they want is and when it will be delivered to the customer.

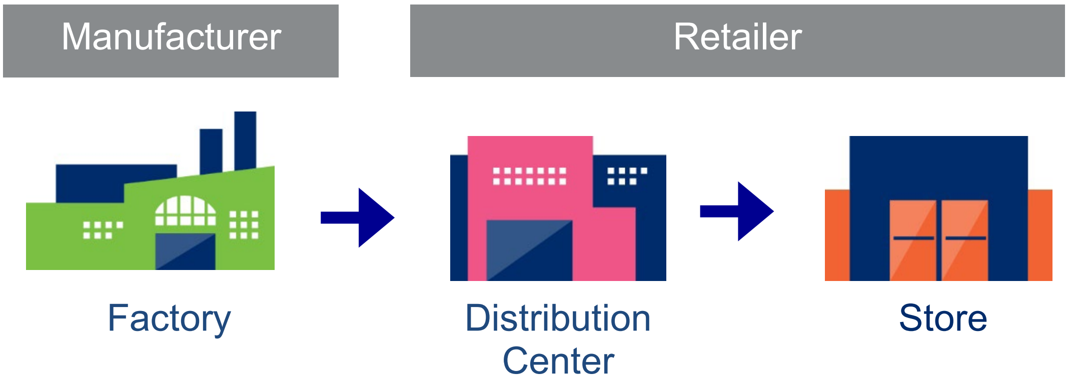
Determining the goal and then clearly documenting the requirements to meet the goal is the first step in beginning to think about how to deploy EPCIS. For example, if an organisation is trying to meet a track and trace regulation, it needs to understand what data is required, at which points in the process, where to keep the data, and who and how the data is being sent to another party. Ponce the overall requirements are understood, the detailed process flow and specific data requirements based on EPCIS and the CBV can be determined.

4.2 Step 2: Document the Business Process flow

Let's take a look at a simplified forward logistic business flow. We will use this business flow in the following sections to illustrate the other steps in the design process.

In this business process we have a manufacturer who is **manufacturing** goods at his production facility. From the manufacturer's factory, the goods are then shipped to the **retailer's distribution centre** where they are received and stored. From the retailer's distribution centre the goods are then shipped to the **retail store** where they are received and sold to the consumer.

Figure 4-1 Example Business Process Flow



The overall business process flow is as follows:

1. The goods are manufactured, and a product is packaged into cases which are in turn packed onto pallets.
2. The products are shipped by truck from the manufacturer's factory to the retailer's distribution centre.
3. The products arrive at the retailer's distribution centre and are received into inventory.
4. The products are shipped from the retailer's distribution centre by truck to the retail store.
5. The products arrive at the retail store and are received into the stockroom.
6. The products are moved from the stockroom to the sales floor.
7. In the retail store the product will be sold to the consumer.

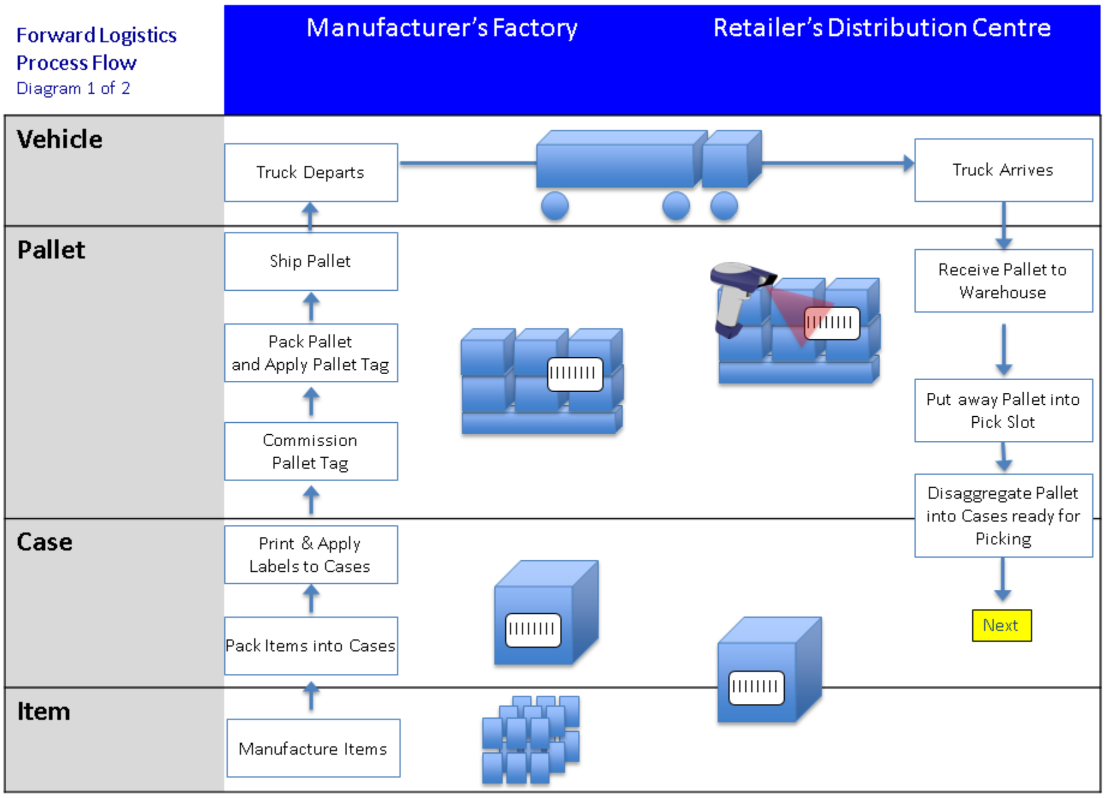
4.3 Step 3: Break each process flow into a series of discrete business steps

The process flow of the simplified forward logistics example is shown in the following diagrams. The blue arrows show the flow, and the white rectangles each represent a single step in the process. As time moves from left to right, the horizontal axis also shows the locations involved as the product moves from one location to another.

In this example, there is an aggregation hierarchy where items are packed into cases, cases are packed into pallets, and pallets are loaded onto trucks. In such cases, it is often helpful to use the vertical axis to show at which hierarchy level each step takes place. If a process flow only works at a single level of aggregation, the corresponding diagram might be completely horizontal, or the vertical axis could be used to highlight some other aspect of the flow. At this stage, the idea is to be as clear as possible about the individual steps of the flow.

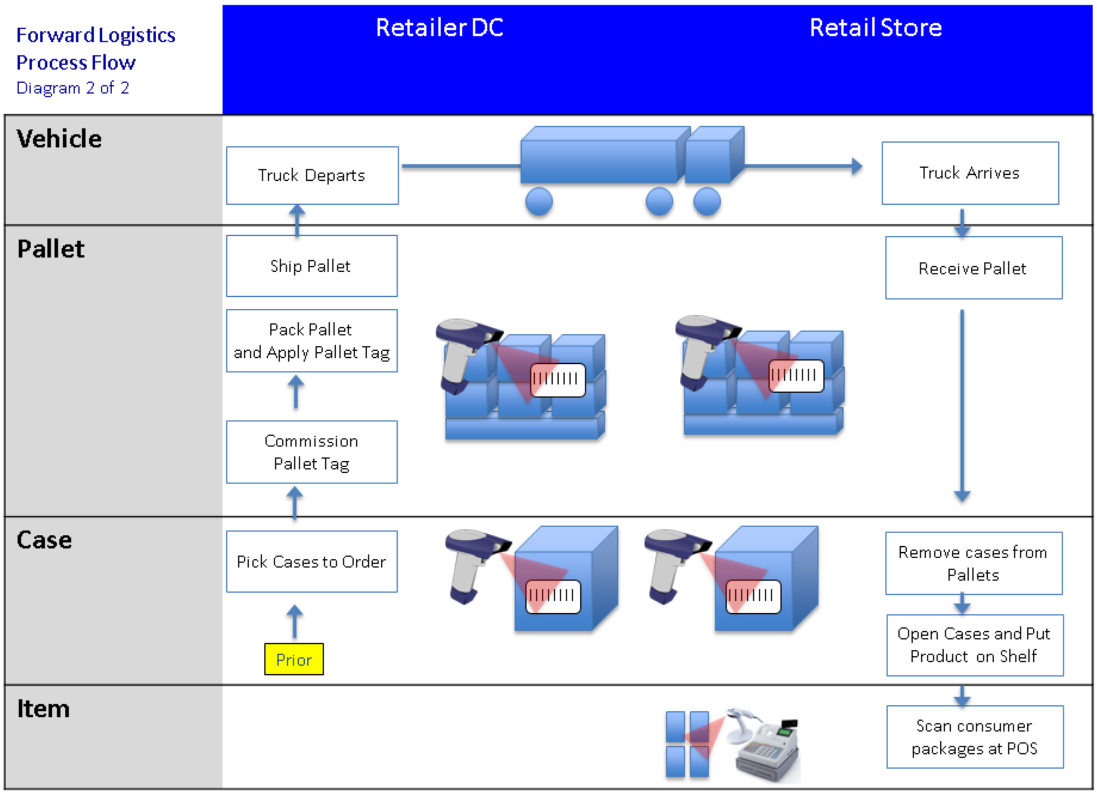
Not every step in these flow charts will lead to an EPCIS event; that is addressed in the next section.

727 **Figure 4-2** Forward Logistics Process Flow, Diagram 1 of 2


Forward Logistics
Process Flow
Diagram 1 of 2
Manufacturer's Factory
Retailer's Distribution Centre
Vehicle
Pallet
Case
Item
Truck Departs
Truck Arrives
Ship Pallet
Receive Pallet to Warehouse
Pack Pallet and Apply Pallet Tag
Put away Pallet into Pick Slot
Commission Pallet Tag
Disaggregate Pallet into Cases ready for Picking
Print & Apply Labels to Cases
Next
Pack Items into Cases
Manufacture Items

728

729 **Figure 4-3** Forward Logistics Process Flow, Diagram 1 of 2



730

4.4 Step 4: Decide which business steps require visibility events

Not every business step in a business process requires a visibility event. The decision about whether a given business step needs an event is typically a trade-off between what data is valuable to have and what data is feasible to collect.

Questions about what data is valuable to have include:

- Will having detailed visibility event information about this step of the process provide useful input to some business application?
- Is information about this step of the process required in order for an application to understand information about another step? For example, if an event at the "shipping" step includes a pallet ID, it might also be necessary to capture an earlier event at the "packing" step so that an application knows the content of the shipped pallet.
- Is information about this step of the process required by a trading partner or by a government regulation?

Questions about what data is feasible to collect include:

- Do the physical or digital objects involved in this step of the process have suitable identifiers? If not, is it feasible to give them identifiers?
- For physical objects, is it feasible to affix the identifiers using a data carrier such as an RFID tag or bar code? If not, will it be possible to capture the identifier some other way?
- Is it feasible to modify the operational process to include data capture of the visibility event? Considerations here include the cost of the necessary infrastructure (bar code scanners, RFID readers, software, etc.) and the impact on process itself (is additional labour needed, will the process slow down, etc.).

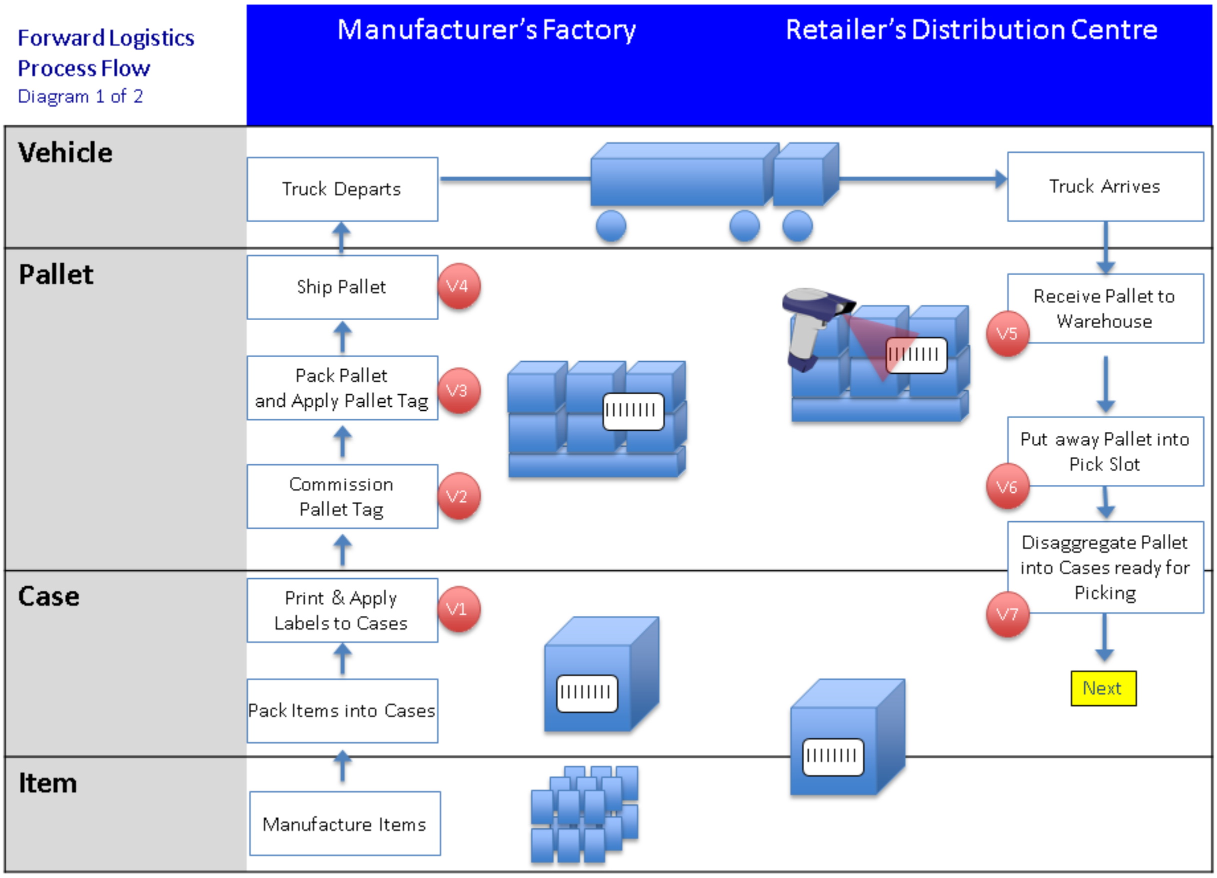
In the example, we will assume that from a business perspective it is essential to know what is shipped and received at each location. In many cases, it is also necessary to have a record of what is "commissioned"; that is, to capture an event each time a new identifier is created. But we will also assume that it is only feasible to capture data at the case and pallet level, not at the item level. We will also assume that the trucks used to move the pallets do not carry identification, and that it would not be feasible to track which trucks are used anyway.

Putting that together, this leads to capturing visibility events at the following steps in the Manufacturer’s portion of the example:

- **V1**: Print and apply case label (commissioning – needed so that later steps are understandable)
- **V2**: Print the pallet label (commissioning – needed so that later steps are understandable)
- **V3**: Pack cases into pallet (needed so that the content of the shipment can be inferred from reading just the pallet identifier)
- **V4**: Ship the pallet

These are indicated in the diagram with red circles numbered V1, V2, etc. Other steps in the diagram not carrying circles are steps for which no visibility events are captured.

Figure 4-4 Forward Logistics Process Flow with Visibility Capture Indicated



4.5 Step 5: Model the completion of each step as a visibility event

Now we begin to design the EPCIS data that will capture what happens in the selected steps of the business process. The first step is to decide what event type best fits the situation at hand, from the list of event types as described in section 3.6. The event type will determine the structure of the information in the *What* dimension of the event.

To choose the event type, consider what physical or digital objects are involved in the event and how they relate to each other. Most often, you will choose one of the following three event types:

- **ObjectEvent**: Use this if there were one or more objects involved in your event, and all the objects participated in the event in the same way. This is by far the most common event type.
- **AggregationEvent**: Use this if your event involves a physical aggregation involving a "parent" object and one or more "child" objects. An example of an aggregation is 12 items (the "children") packed into a carton (the "parent"). Other examples of aggregation include cases on a pallet, items in a tote, cartons loaded into a truck, containers loaded onto an ocean vessel, and components installed in an assembly. In all of these examples, each child retains its identity even while aggregated to the parent, and the aggregation is reversible (that is, it may be "disaggregated").
- **TransformationEvent**: Use this if your event is a process in which one or more "input" objects are consumed and one or more "output" objects are produced. Unlike an aggregation, where the can later be separated from a parent, in a transformation the input objects no longer exist after the event. Examples of transformations include mixing raw materials to create a finished recipe, repackaging items such that the original package no longer exists and a new GTIN labels the new package, and smoking salmon to transform raw fish into smoked fish.

The fourth event type, the **TransactionEvent**, can be used if your event if a process in which one or more objects are definitively associated with (or disassociated from) one or more business transactions. However, because business transactions can be included in the *Why* dimension of all the other event types, there is seldom a need to use the TransactionEvent type.

The ObjectEvent and AggregationEvent types have an additional qualifier, the *action*, which says how the event relates to the lifecycle of the object and the aggregation, respectively. Specifically:

- For an **ObjectEvent** the action values are:
  - ADD if the event marks the beginning of the life of the object. No other events for the same objects should precede this one. This is most often used when the business step is "commissioning."
  - DELETE if the event marks the end of the life of the object. No other events for the same objects should follow this one. This is most often used when the business step is an end-of-life step such as "decommissioning," "destroying," or a business step involving sale to a consumer (if there is no possibility to track the object post-sale).
  - OBSERVE in all other cases.
- For an **AggregationEvent** the action values are:
  - ADD if children are added to the aggregation during the event; e.g., when packing items into a case.
  - DELETE if children are removed from the aggregation during the event; e.g., when unpacking items from a case.
  - OBSERVE if the parent and children are in a state of aggregation during the event but no children are added or removed.

The **TransactionEvent** also has an action qualifier; see the EPCIS standard for details. The **TransformationEvent** does not have an action qualifier.

Here is how event types would be assigned to events V1 through V4 of the example from the previous section:

Table 4-1 Assignment of Event Types to Business Process Steps in Example Business Process

Event	Description	Event Type	Comment
V1	Print and apply case label	ObjectEvent ADD	This is the beginning of life for the SGTIN that identifies the case
V2	Print and apply pallet label	ObjectEvent ADD	This is the beginning of life for the SSCC that identifies the pallet
V3	Pack cases onto pallet	AggregationEvent ADD	Children (the cases) are added to the aggregation
V4	Ship pallet	ObjectEvent OBSERVE or AggregationEvent OBSERVE	See discussion below

In the V4 event, there is a choice in how to record the act of shipping the pallet as an EPCIS event. One approach is to use an **ObjectEvent** (with action OBSERVE) and include only the SSCC of the pallet in the *What* dimension. This makes the data capture easier, and results in a more compact event, but it means that applications receiving the data will need to consult the V3 event too if they need to infer what cases were on the pallet that was shipped. An alternative approach is to use an **AggregationEvent** (with action OBSERVE) and include both the SSCC of the pallet (the parent) and the SGTINs of all the cases (the children) in the *What* dimension. This approach makes sense if it is feasible to know the case SGTINs at the time the pallet is shipped, and if the Manufacturer wishes to be explicit about exactly which cases are on the pallet at that time. Applications receiving V4 would not need to make any inferences using V3 to know what cases are on the pallet.

The V4 example illustrates the subtle choices that sometimes must be made in deciding how to model business processes using EPCIS. To assist in such situations, it is helpful to consult industry sector-specific guidelines that provide standard EPCIS models for business processes commonly arising in those sectors.

4.6 Step 6: Decide what data fields are to be included in the visibility event

Once the basic event types are decided upon, the next task is to decide what data to include in the *What*, *When*, *Where*, and *Why* dimensions of each event. It is tempting to approach this from the perspective of what information is available to your capturing application, such as what data comes out of an RFID reader or bar code scanner. However, EPCIS data is much more useful if you approach it from the opposite direction, namely from the perspective of a business application consuming the data. The question to ask yourself is: "what information does a business application need to understand what happened during this event?" The business application doesn't need to know *how* the data was captured; it needs to know *what* happened from a business perspective.

A good way to proceed is to consider each of the four data dimensions in turn.

4.6.1 Designing the What Dimension

The *What* dimension identifies the physical or digital objects involved in the event. The structure of the information in the *What* dimension depends on the event type:

- For an **ObjectEvent**, the *What* dimension contains a list of objects. All objects participate in the event in the same way.
- For an **AggregationEvent**, the *What* dimension names a specific object as the "parent" and contains a list of other objects as the "children." (There are two exceptions. If the action is OBSERVE the parent may be omitted, indicating that the children were observed in a state of aggregation but the identity of the parent is unknown. If the action is DELETE the children may be omitted, indicating that *all* children are disaggregated from the parent.)
- For a **TransformationEvent**, the *What* dimension includes one list of objects that are the inputs to the transformation, and a second list of (different) objects that are the outputs of the transformation. (If a TransformationEvent is connected to other TransformationEvents through the TransformationID, it may omit either the inputs or the outputs; see section 5.5.2.)

Besides considering which objects involved the business process step are relevant to the event, you also have to determine how those objects will be named in the event. In EPCIS there are two different ways to refer to an object:

- **Instance-level Identification**: If an object has an identifier that is unique to that particular object, it is called instance-level identification. Examples of instance-level identification include a Global Trade Item Number (GTIN) with a serial number (together called a Serialised GTIN, or SGTIN), a Serial Shipping Container Code (SSCC), a Global Returnable Asset Identifier (GRAI) that includes a serial number, and so on.
- **Class-level Identification**: If an object has an identifier that is identical to the identifier carried by other, similar objects, it is called class-level identification. Examples of class-level identification include a GTIN plus a batch or lot number (shared by all trade items belonging to the same batch or lot), a GTIN by itself, a GRAI without a serial number, and so on.

Instance-level identification is the most powerful in terms of how EPCIS data can be used by applications, because instance-level identification makes it possible to recognise that an object referenced in one event is the *very same object* as an object referenced in a prior or subsequent event. On the other hand, assigning instance-level identification to objects is usually a more complex business process than assigning class-level identification.

When class-level identification is used there may be more than one object involved in the event from the same class, so a class-level identifier is usually accompanied by information that specifies the quantity. Including instance-level identification, this results in four ways an object could be identified in the *What* dimension of an EPCIS event:

Table 4-2 Class and Instance Level Object Identification

Instance- or Class-level	*What* Dimension Contents	Meaning
Instance	An instance-level identifier (SGTIN, SSCC, GRAI with serial, etc.)	A specific object participated in the event

Instance- or Class-level	*What* Dimension Contents	Meaning
Class	A class-level identifier (GTIN, GTIN+Lot, GRAI without serial, etc.) plus an integer quantity	A specific number of objects belonging to the specified class participated in the event. The class in this case refers to discrete objects that can be counted.
	A class-level identifier (GTIN, GTIN+Lot, GRAI without serial, etc.) plus a real amount and unit of measure	A quantity equal to the specified physical measure (amount + unit of measure) of the specified class participated in the event. The class in this case refers to objects that must be measured rather than counted, such as liquid dispensed in arbitrary volumes or solids dispensed in arbitrary weights.
	A class-level identifier (GTIN, GTIN+Lot, GRAI without serial, etc.), with no quantity information	Some unspecified quantity or amount of the specified class participated in the event.

The last case in the table, a class-level identifier with no quantity information, should only be used rarely, when it is impossible to determine the quantity or if the quantity is to be withheld for privacy reasons.

The same EPCIS event might have some objects identified using instance-level identification and others identified using class-level identification. For example, cases identified by GTIN and lot (class-level) could be aggregated to a pallet identified by SSCC (instance-level), or there could be a transformation event where some inputs are raw materials identified by class and quantity, other inputs are identified by GTIN+serial number (instance-level), and the outputs are identified by GTIN+serial number. However, a *given* object should only be identified one way in an event. For example, if an object event has five SGTINs which are different serial numbers for the same GTIN, the object event should include those five SGTINs but *not* also include the GTIN as a class-level identifier.

4.6.2 Designing the When Dimension

The *When* dimension is the most straightforward of the four dimensions. It is required in every event, and always contains two pieces of information:

- **EventTime**: The date and time at which the event occurred. This is always expressed in a format that includes a time zone specifier, so that it unambiguously identifies a moment in time.
- **EventTimeZoneOffset**: The time zone offset (relative to UTC) that was in effect at the place where the event took place. This allows the *EventTime* to be displayed to users in the local time where the event happened, if desired.

The correct value to use for these two data elements is usually quite obvious and so there is little design work to be done.

For a business step that takes place over a long interval of time, there may be some question as to whether *EventTime* should be the moment when the step begins or ends, or some moment in between. Usually, the ending time of the business step is the most appropriate. But as with all EPCIS data design questions, it should be considered from the perspective of a business application consuming the data. If it is important to business applications to know both the starting time and the ending time of a business step, you should consider whether it would be more appropriate to model the process using *two* EPCIS events, one for the start of the process and one for the end.

Conversely, sometimes there are several different events from a business perspective which are carried out simultaneously or in a way that would make it difficult to assign a different *EventTime* for each. For example, an automated manufacturing machine might assign SGTINs to twelve products ("commissioning" business step), assign another SGTIN to a case ("commissioning" again), and pack the items into the case ("packing" business step), all at once. It may not be physically all at once, but the EPCIS Capturing Application built into the machine may not have any way to distinguish the times. In such cases it may be appropriate to assign the identical event time to all EPCIS events generated, but if there is a logical sequencing of the events it is usually much better for consuming applications if the event times are slightly altered so that the chronological order is logical. In the items-into-case example, the EPCIS event for "packing" (the aggregation event) should have an event time that is later than the commissioning events, even if it is artificially set to a time only one millisecond later. This allows consuming applications to order the events by their *EventTime* to arrive at a logical sequence.

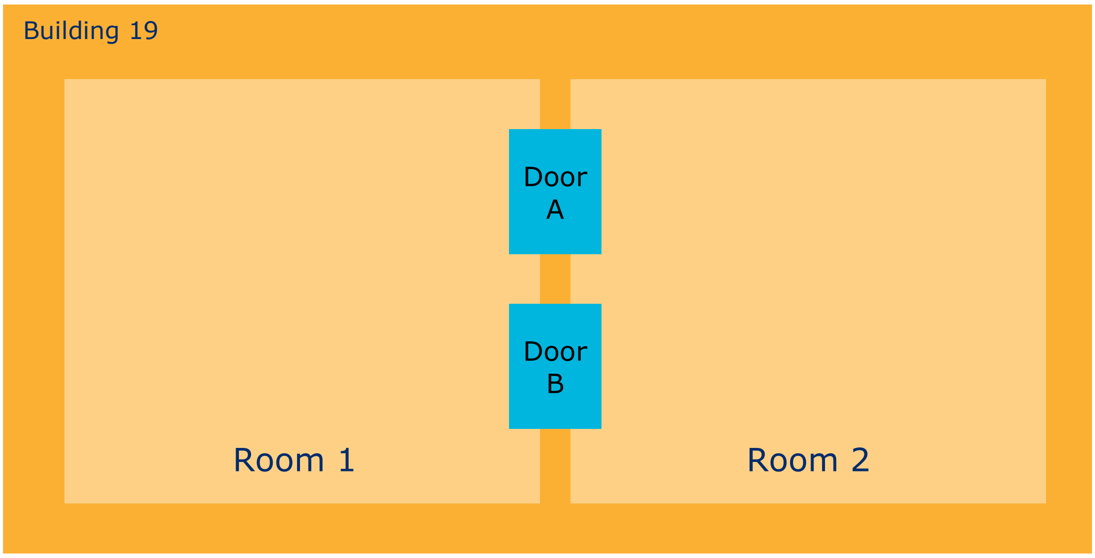
4.6.3 Designing the Where Dimension

The *Where* dimension identifies the physical location of objects in the event. The two data elements in the *Where* dimension are both optional, but most EPCIS events will include them. The two data elements are:

- **ReadPoint**: The *ReadPoint* identifies where the objects named in the *What* dimension were at the time of the event; that is, where the event took place.
- **BusinessLocation**: The *BusinessLocation* identifies where the objects named in the *What* dimension are expected to be following the event, until another event says otherwise.

The names *ReadPoint* and *BusinessLocation* can be a little confusing. For example, the *ReadPoint* could be just as relevant from a business perspective, or more so, than the *BusinessLocation*, depending on the situation. Instead of trying to read meanings into the names "read point" and "business location", just remember the definitions: *ReadPoint* is the location of the objects at the time of the event, *BusinessLocation* is the location afterwards.

The difference between *ReadPoint* and *BusinessLocation* can be visualised by imagining a facility having several rooms connected by doorways, like this:



Imagine that an EPCIS event is captured whenever an object moves through one of the doorways; e.g., by an RFID reader stationed at each door. Imagine an object moves from Room 1 to Room 2 by passing through Door A. In this case, the *ReadPoint* (the location at the time of the event) is Door A and the *BusinessLocation* (the location afterward) is Room 2. Note that the object might move around within Room 2 without generating any new EPCIS events, so at the time it moved into Room 2 all we know about where it is afterwards is that it is somewhere within Room 2. If instead the object had moved from Room 1 to Room 2 via Door B the *BusinessLocation* would still be Room 2 but the *ReadPoint* would be Door B. On the other hand, if the object later moves in the opposite direction through Door A the *ReadPoint* would again be Door A but the *BusinessLocation* would be Room 1.

The reason it is useful to have *BusinessLocation* is that it helps to answer the question "where is the object *right now*?" If you happen to ask that question right at the moment an event takes place then the *ReadPoint* tells you that, but at any other time the *BusinessLocation* of the most recent event is the best available approximation to location of the object right now. At the same time, *ReadPoint* is useful because it tells you something about the past: "where was the object when X happened to it?" (where X is described by the *Why* dimension of the appropriate event).

A key question in designing the *Where* dimension is to decide at what *granularity* you will describe location. For example, if an object enters through a loading dock door during a receiving operation, there are several ways you could describe the location of the event (the *ReadPoint*), listed here from most specific (finest granularity) to least specific (coarsest granularity):

- "Receiving Dock #5 in Building 2 of the Chicago campus of XYZ company"

- "The receiving area in Building 2 of the Chicago campus of XYZ company" (specific door not specified)
- "Building 2 of the Chicago campus of XYZ company" (specific area within Building 2 not specified)
- "The Chicago campus of XYZ company" (specific building not specified)
- "XYZ company" (specific location not specified)

Deciding what level of granularity to include in an EPCIS event is a key decision. As with most EPCIS design decisions, it will be a trade-off between what business applications *need* to make use of the data and what it is feasible to *collect* when the EPCIS event is captured. For example, distinguishing between different loading dock doors in a building may require more expensive infrastructure than just knowing that an object has entered the building. At the same time, it might not be important for business applications to know what specific door was used. Sometimes, the question of granularity is answered differently in designing EPCIS events as they are *captured* internally versus how they are *shared* with trading partners. For example, the *ReadPoint* might be captured internally at the level of individual loading dock doors, but then redacted to the "building" level when sharing the same event with a trading partner. (See also section 6.7.)

It is common for the *BusinessLocation* to be expressed at a coarser level of granularity than the *ReadPoint*, simply because an EPCIS capturing application has less certainty about where an object might be following an event compared to where it is at the moment the event takes place. It is also common for *ReadPoint* to be at the same level of granularity as *BusinessLocation* when there is no business need to express *ReadPoint* with any finer precision.

A special case for *BusinessLocation* occurs when objects are transferred from one location to another, as in shipping followed by receiving. When the object is shipped, the location of the objects following the event is obviously not the location of the shipper. But it is also not the location of the receiver, because it is only after the event captured during receiving that the object is located at the receiver. Therefore, the correct *BusinessLocation* for the EPCIS event captured at shipping is "unknown" – at the time of shipping, it is unknown where the object will be until the receiving operation takes place. This is expressed in EPCIS by omitting the *BusinessLocation* data element entirely from the shipping event.

4.6.4 Designing the Why Dimension

The *Why* dimension explains the business context for the event, and is crucial for business applications to make sense of EPCIS data. All of the data elements in the *Why* dimension are optional, but almost all EPCIS events will include at least the *BusinessStep* and *BusinessLocation* data elements. The other data elements in the *Why* dimension are included only when they are relevant to the business step being carried out.

The definitions of the data elements in the *Why* dimension were given in section 3.5. Here are design considerations for choosing whether to include each data element, and how to choose the appropriate values.

4.6.4.1 Designing the Business Step

The *BusinessStep* data element is the most important when it comes to a business application understanding what EPCIS data means. The *BusinessStep* value is an identifier that says what step of the business process was taking place at the time of the event. Without the business step an application only knows that an object existed at a particular place ant time; with the business step an application knows how that object relates to the overall business. Practically all EPCIS events should have a *BusinessStep* value. The *BusinessStep* value usually corresponds to a verb of some kind: shipping, receiving, packing, etc.

In order for business step values to be useful, they must have a meaning that is known in advance to the applications that will see them. For this reason, the value for *BusinessStep* is always defined by a standard of some kind – a document that maps a given *BusinessStep* value to an explanation of what the value means and how to interpret the EPCIS event carrying that value. The CBV is one such standard. It is a global standard that defines several dozen business step values that apply to a variety of business steps commonly occurring in supply chain business processes across many industry sectors. Because it is a global, cross-sector standard, using CBV business step values makes an EPCIS event intelligible to the widest set of applications. When a CBV business step value is applicable, it should be used.

Sometimes, however, you may be using EPCIS in a business process that includes a step that does not fit very well with any of the business step values defined in the CBV. In such cases, a different identifier must be used, one that you create yourself for the specific application. There will still be a document that defines the identifier and its meaning – in this case the document is an internal design document rather than a global standard. Specific business step values may also be defined across a group of trading partners, or by a sector-specific standard. However, all such values will result in EPCIS events that can only be understood within the smaller group of organisations that is aware of the narrower standard or design document that defines them. This is a trade-off that must be considered when deciding whether to use the CBV or not.

Section 4.7 describes *how* to create an identifier not defined in the CBV so as to avoid conflicts.

4.6.4.2 Designing the Disposition

The *Disposition* value is an identifier that indicates the business condition of the objects following the event. The *Disposition* value usually corresponds to an adjective that describes the business state of the objects as it relates to the overall business process: in_progress, recalled, damaged, etc.

A key use of the *Disposition* is to note the difference between normal flow and exceptions. For example, the CBV disposition value "`in_progress`" indicates objects that are moving normally through the supply chain and "`recalled`" indicates objects that have been recalled to the manufacturer. Having a *Disposition* separate from *BusinessStep* helps model such situations in two ways. One, at the time of an event that is subject to exceptional outcomes, the *Disposition* can express which outcome occurred. For example, there may be an EPCIS event with *BusinessStep* "inspecting" (from the CBV) where the outcome of the inspection is either *Disposition* "`in_progress`" in the usual case or "`recalled`" if the inspection discovers the object is subject to recall. Two, the *Disposition* can continue to indicate the exceptional state even as the objects are subjected to further events. For example, following the "inspecting" step a recalled object might have several EPCIS events with *BusinessStep* values "`shipping`" and "`receiving`" as the object works its way upstream to the manufacturer. Without *Disposition* these EPCIS events would be difficult to distinguish from ordinary shipping and receiving steps, but with a *Disposition* value of "`recalled`" instead of "`in_progress`" it becomes clear that these events are part of a reverse logistics process.

As with *BusinessStep*, values of *Disposition* are only useful if their meaning is known in advance to the applications that will see them. For this reason, all of the comments in section 4.6.4.1 apply equally to *Disposition* values.

4.6.4.3 Designing the Business Transaction List

The *BusinessTransactionList* is a list of references to business transactions – data that are available from other systems besides EPCIS. Examples of a business transaction include: a reference to a specific purchase order, a reference to a specific invoice, and so forth. This information provides business context for an EPCIS event and helps link EPCIS data with other business information systems.

Each business transaction in the *BusinessTransactionList* consists of a pair of identifiers. The first is the business transaction type identifier, which says what kind of business transaction is being referenced (purchase order, invoice, etc.). The second is the business transaction identifier that references the specific transaction of the specified type.

Business transaction type identifiers are similar to *BusinessStep* or *Disposition* values in that they are useful only if their meaning is known in advance to the applications that will see them. For this reason, all of the comments in section 4.6.4.1 apply equally to business transaction type values. The CBV such as purchase order, invoice, etc.

The second part of a business transaction reference, the business transaction identifier, refers to a specific business transaction. Unlike business step, disposition, or business transaction type values there is not a fixed list of business transaction identifiers – new identifiers are constantly created as new business transactions are created. Typically, a business transaction identifier is generated by some information system other than EPCIS; for example, an invoice number might be created by an Enterprise Resource Planning (ERP) system.

A business transaction identifier must be globally unique in order to be used in an EPCIS event. This is because in processing EPCIS data an application might gather EPCIS events from across the supply chain. In that situation, it is essential that two purchase orders from different parties in the supply chain cannot be confused.

There are two strategies for creating globally unique business transaction identifiers suitable for use in an EPCIS business transaction list. One is for the system creating the business transaction to use a globally unique identifier as the only way it refers to the transaction. For example, an ERP system might natively assign a unique identifier such as a GS1 Global Document Type Identifier (GDTI). If assigned correctly, a GDTI issued by one system will be different than a GDTI generated by any other party's system. Many legacy systems, however, are not designed to do this – a typical ERP system will simply give each transaction a number like 12345, which is unique within the context of that ERP system but not guaranteed to be unique compared to the numbers generated by another ERP system.

The second strategy for creating a globally unique business transaction identifier is to combine the identifier created by a legacy system with a prefix that makes it globally unique. The CBV specifies a template that may be used for this purpose which uses the Global Location Number (GLN) of the issuing party. For example, if Company X has a party GLN of 0614141123452 and its ERP system issues purchase order #12345, the corresponding globally unique identifier using the CBV template is:

`urn:epcglobal:cbv:bt:0614141123452:12345`

The first part of this identifier, `urn:epcglobal:cbv:bt:`, is a prefix indicating that the CBV's business transaction identifier template is used. The remaining two components are the GLN and the PO number assigned by the ERP system, respectively. The entire string considered as a single identifier is globally unique, because PO #12345 from any other ERP system would be given a different prefix. (If one company has multiple ERP systems, and there is the possibility that their assigned transaction numbers will collide, a different GLN should be used as the prefix for each system.)

When processing EPCIS data, the entire business transaction identifier, including any prefixes, should be used. For example, to test whether two EPCIS events make reference to the same business transaction, the entire identifier strings should be compared (along with the business transaction type identifiers). However, when relating EPCIS data to legacy system data, it may be necessary to recognise the CBV prefix and parse the identifier to identify which legacy system is referred to and what is the native transaction ID for that system.

4.6.4.4 Designing the Source and Destination Lists

Certain business process steps are part of a process of *business transfer* where ownership and/or physical possession passes from one party to another. Shipping and receiving are two common examples, but there may be others such as consigning, accepting, returning, intermediate transportation steps, and so on. In such cases it is often useful to include information that identifies both ends of the transfer. For example, in a shipping event it is useful not only to indicate the "ship from" location but also the "ship to" location. It may also be useful to indicate the parties involved at both ends, both from an ownership perspective as well as a physical possession perspective, which may or may not be the same pair of parties. The source and destination lists in an EPCIS event may be used to provide this information. Source and destination information is part of the *why* dimension of an EPCIS event, as it serves to provide business context.

The source list consists of a list of sources, each of which is a pair consisting of a source type and a source identifier. Likewise, the destination list consists of a list of destinations, each a pair of a destination type and a destination identifier. There are three possible source or destination types defined in the CBV; each says how to interpret the source or destination identifier that it qualifies:

Table 4-3 Source/Destination Types Defined in the CBV

Source or Destination Type	Meaning
`owning_party`	The source or destination identifier denotes the party who owns (or is intended to own) the objects at the originating endpoint or terminating endpoint (respectively) of the business transfer of which this EPCIS event is a part.
`possessing_party`	The source or destination identifier denotes the party who has (or is intended to have) physical possession of the objects at the originating endpoint or terminating endpoint (respectively) of the business transfer of which this EPCIS event is a part
`location`	The source or destination identifier denotes the physical location of the originating endpoint or terminating endpoint (respectively) of the business transfer of which this EPCIS event is a part

The source or destination identifier itself is a globally unique identifier for a party or physical location, depending on the source/destination type. Often this is a GLN (with or without extension) or PGLN (Party GLN), but the CBV also specifies other identifiers that could be used.

Any combination of the three source/destination types may be used in either the source list or destination list or both, according to what business context is available. Typically, both a source and a destination of a given type are included.

A complete business transfer typically extends across multiple EPCIS events, often generated by more than one party. For example, a very simple transfer would include one EPCIS event for the shipping step and a second EPCIS event for the receiving step. A more complex transfer might involve separate arriving and accepting steps, for example, or tracks intermediate in-transit steps such as observing a rail carrier or ocean carrier during its passage. All such steps belonging to the same transfer could include source/destination information. When this is the case, the source/destination information is usually the same on all events.

For example, in a transfer of possession from Party A to Party B, both the shipping and receiving EPCIS events could include a source of type "possessing party" for Party A and a destination of type "possessing party" for Party B. The interpretation of the source/destination information on the two events is subtly different. In the shipping event, the source indicates the *known* possessing party at the origination of the transfer but the destination indicates the *intended* possessing party at the termination of the transfer. In the receiving event, the destination indicates the *known* possessing party at the termination of the transfer and the source indicates the *believed* possessing party at the origination of the transfer.

A source or destination of type "location" may coincide with read point information in the *where* dimension for certain events. Specifically, the read point in a shipping (or similar) step coincides with the source of type "location," and the read point in a receiving (or similar) step coincides with the destination of type "location." In such cases, the information in the source/destination should be consistent with the information in the read point. (It might not be identical if, for example, the read point is reported using a more granular location identifier than the source or destination.)

An EPCIS event that is not part of a business transfer should not include source/destination information.

See section 5.2 for an example scenario that uses the source/destination list.

4.6.5 Example

Putting together all of the material in this section, let's illustrate how we would design the EPCIS event for Event V4 of the example from section 4.4. In this event, a pallet containing several cases is shipped from the Manufacturer to the Retailer's Distribution Center.

As noted in section 4.5, this event could be represented as an **ObjectEvent** naming just the pallet, or as an **AggregationEvent** naming both the pallet and the cases. We will assume the **ObjectEvent** approach in this illustration.

Table 4-4 EPCIS Event Information Content for Step V4 of Example From section 4.4

Dim	Data Element	Design Choice	Comments
	Event Type	Object Event	See above
	Action	OBSERVE	This is neither the beginning of life nor the end of life for the pallet, so the action is OBSERVE (see section 4.5).
What	**EPC List**	A list containing one element: the SSCC of the pallet (instance-level identification)	
When	**Event Time**	The date and time at which the pallet is shipped	
	Event Time Zone Offset	The time zone offset in effect where the pallet was shipped	Local time is five hours earlier than UTC
Where	**Read Point**	Shipping dock #2 of building 10	In this case, we have chosen to capture the read point at a very fine level of granularity
	Business Location	(omitted)	As noted in section 4.6.3, the business location is omitted for a shipping event because we don't know where the pallet will be until a subsequent event takes place during receiving.
Why	**Business Step**	Shipping (from CBV)	A standard identifier defined in CBV 1.1 ensures that all consuming applications will understand this event
	Disposition	In Transit (from CBV)	A standard identifier defined in CBV 1.1 ensures that all consuming applications will understand this event. "In Transit" indicates normal forward progress during a transfer from shipper to receiver.
	Business Transaction List	A list containing two business transaction references: the Retailer's purchase order and the Manufacturer's invoice.	"Purchase Order" and "Invoice" are standard identifiers defined in CBV 1.1 to identify business transaction types.
	Source List	A list containing one source of type "owning party," indicating the Manufacturer as the owning party at the source	Shipping is a step within an overall transfer of ownership from source to destination. Here, the owning party at the source (the shipper) is identified. "owning_party" is a standard identifier defined in the CBV to identify a type of source
	Destination List	A list containing one source of type "owning party," indicating the Retailer as the intended owning party at the destination	Shipping is a step within an overall transfer of ownership from source to destination. Here, the intended owning party at the destination (the shipper) is identified. "owning party" is a standard identifier defined in the CBV to identify a type of source

4.7 Step 7: Determine the Vocabularies that populate each Data Field

In the previous step, you determined what you want the data elements of each EPCIS event to say. The next step is to translate the informal description of each data element's contents into a specific identifier that a computer can understand. The place to start is sections 7 and 8 of the CBV.

4.7.1 Vocabularies for the What dimension

In the *What* dimension, you have references to one or more physical or digital objects. Most of the time, each object will be identified by a GS1 Key. For example, a trade item might be identified by a GTIN (example: 00614141123452) and a serial number (example: 400). In EPCIS, the GTIN plus serial number is represented either as:

- (for EPCIS 1.x and 2.0) an **EPC "Pure Identity" URN**, normatively specified in GS1's EPC Tag Data Standard [TDS], e.g.:

```
urn:epc:id:sgtin:9521141.012345.400
```

or

- (for EPCIS 2. 0 and later) a **GS1 Digital Link URI**, normatively specified in the GS1 Digital Link Standard: URI Syntax [GS1DL], e.g,:

```
https://example.org/01/09521141123454/21/400
```

New deployments of EPCIS are strongly encouraged to use of GS1 Digital Link URIs, due to their native interoperability with GS1 element strings.

4.7.2 Vocabularies for the Where dimension

The *ReadPoint* and *BusinessLocation* data elements in the *Where* dimension contain identifiers that refer to physical locations. To choose an appropriate identifier, you must first decide how locations will be identified.

The most common way to identify a location is to give it a unique identifier such as a Global Location Number (GLN). A GLN is just an arbitrary number that the owner of a location designates to refer to a specific location. A GLN can be assigned at any level of granularity (see section 4.6.3), and you can even assign a GLN to a fine-grain location such as a room in a building and also assign a different GLN to a coarse-grain location such as the building itself. When this is done, GLNs fall into a hierarchy.

When assigning identifiers to very fine-grain location such as individual loading dock doors or individual bins in a large warehouse, the GLN by itself does not have sufficient capacity. In such situations each location can be assigned a GLN plus a GLN extension. When a GLN+extension is assigned to a fine-grain location, the GLN part is usually the GLN of a coarser-grained containing location, such as the containing building.

As in the *What* dimension, the *Where* dimension uses either EPC "Pure Identity" URNs (EPCIS 1.x and 2.0) or GS1 Digital Link URIs (EPCIS 2. 0 and later) to express GS1 identifiers. For example, suppose a location is identified by GLN 9521141111116 and extension 987. In EPCIS, the GLN+extension is represented either as:

- (for EPCIS 1.x and 2.0) an **EPC "Pure Identity" URN**, normatively specified in GS1's EPC Tag Data Standard [TDS], e.g.:

```
urn:epc:id:sgln:9521141.11111.978
```

or

- (for EPCIS 2.0 and later) a **GS1 Digital Link URI**, normatively specified in the GS1 Digital Link Standard: URI Syntax [GS1DL], e.g.:

```
https://example.org/414/9521141111116/254/978
```

New deployments of EPCIS are strongly encouraged to use of GS1 Digital Link URIs, due to their native interoperability with GS1 element strings.

To represent a **GLN without an extension**,

- (for EPCIS 1.x and 2.0) an **EPC "Pure Identity" URN**, normatively specified in GS1's EPC Tag Data Standard [TDS], e.g.:

```
urn:epc:id:sgln:9521141.11111.0
```

where a single `0` digit is used in place of the extension;

or

- (for EPCIS 2.0 and later) a **GS1 Digital Link URI**, normatively specified in the GS1 Digital Link Standard: URI Syntax [GS1DL], e.g.:

```
https://example.org/414/9521141111116
```

where the GLN extension is omitted from the GS1 Digital Link URI.

Sometimes a location can only be identified by geospatial coordinates—latitude and longitude—rather than by a unique identifier. The most common case for this is as a *ReadPoint* when tracking a vehicle such as an ocean vessel while in transit, where there are no pre-defined locations that could be identified by GLN on the open ocean but a Global Positioning System receiver is available. In this case, a geospatial URI may be used. It looks like this:

```
geo:22.300,-118.44
```

This example denotes the geographic location with latitude 22.300 degrees (north) and longitude 1032 118.44 degrees (west). For more details, see the CBV.

4.7.3 Vocabularies for the Why dimension

The *Why* dimension of an EPCIS event contains many data elements that require identifiers of various kinds. There are two ways this is done depending on the data element.

4.7.3.1 Standard Vocabulary Elements for the Why dimension

Some data elements in the *Why* dimension contain names of concepts that all parties in the supply chain must understand in advance. An example is the *BusinessStep* data element, which contains an identifier representing a concept such as "shipping," "receiving," etc. These identifiers are always defined in a standard of some sort, and the most commonly used standard for this purpose is the CBV.

Section 7.1 of the CBV defines over 30 different business step values.

To select the appropriate business step value, consult the definitions given in the CBV. For example, the CBV defines `packing` to mean "a specific activity within a business process that includes putting objects into a larger container – usually for shipping. Aggregation of one unit to another typically occurs at this point."

In some situations, there is no CBV identifier that is appropriate. In this case, you can create your own identifier, but it should be in URI syntax and use a prefix that is under your control. For most purposes, this means using own your Internet domain name. For example, if you are the Example Corporation with a domain name `example.com` and you need a new business step for "fiddling," you could use a URI like this:

```
http://epcis.example.com/bizstep/fiddling
```

The fact that this begins with `http://epcis.example.com/` means that it will not conflict with a CBV identifier, nor with a private identifier created by any other organisation. If a trade organisation creates a private identifier for a standard it creates, the Internet domain name of the organisation could be used as the root. As noted in section 4.6.4.1, if you create a private business step like this you will have to inform trading partners what it means, so this is less interoperable than using one defined in the CBV.

Note that while the above identifier looks like something you might type into a web browser, as far as EPCIS is concerned it is just an identifier for a business step and there does not have to be a web page accessible via that URI. On the other hand, a web page with that URI might be a very good place to provide documentation for humans about what your business step means.

Several other data elements in the *Why* dimension work the same way; they are summarised below.

Table 4-5 Examples of Standard Vocabulary Identifiers Defined in the CBV

EPCIS Data Element	CBV section	Example
BusinessStep	7.1	`shipping`
Disposition	7.2	`in_transit`
BizTransaction (*type* subfield)	7.3	`po`
Source or *Destination* (*type* subfield)	7.4	`owning_party`

For all of these data elements, the best choice is to use one of the identifiers defined in the CBV, but if this is not possible a private identifier can be constructed as illustrated above.

4.7.3.2 User Vocabulary Elements for the Why dimension

Some data elements in the *Why* dimension identify business objects such as business transactions, sources, destinations, and transformation identifiers. For these data elements, the CBV provides templates that can be used to construct suitable identifiers.

A key consideration here is that identifiers in any dimension of an EPCIS event should be unambiguous. This is especially important when EPCIS events are brought together from across a supply chain. Suppose that the *BusinessTransaction* data element in an EPCIS event in a shipping step contains a reference to a purchase order. It is not sufficient for the EPCIS event to simply say "PO # 1234" because many companies within the supply chain might issue a purchase order with that same number. In an EPCIS event, a reference to a purchase order must be *globally* unique.

The CBV solves this by providing a template for constructing a globally unique identifier.

Some large companies have more than one system that generates purchase orders, e.g. a different system for each division of the company, so there is a possibility of having two purchase orders numbered 1234 from the same company. But this is easily handled by using a different GLN to prefix the PO #s of the two systems; e.g., by using the division-level GLN.

This is one of several ways of constructing globally unique business transaction identifiers defined in the CBV (section 8.5). Another way is to use a GS1 Key such as a GDTI (including serial number). This works if the system that generates the business transaction is already using a GS1 Key as the numbering system. The CBV also shows how to use a private prefix to create business transaction identifiers, though these methods are seldom used.

Advanced use of EPCIS Transformation Events sometimes requires a "Transformation ID" to link together multiple events. Section 8.7 of the CBV describes ways of constructing Transformation IDs, including a GLN-based method similar to the above.

Source and Destination identifiers are described in section 8.6 of the CBV. Most commonly, these are populated with GLNs, just as for location identifiers (section 4.7.2).

4.7.4 Example

Putting together all of the material in this section, here is how the design choices made in section 4.6 would be finally realised as actual identifiers in the EPCIS event.

Table 4-6 Example Assignment of Identifiers for EPCIS Event From section 4.6

Dim	Data Element	Design Choice (section 4.6)	Actual EPCIS Event Contents
	Event Type	Object Event	
	Action	OBSERVE	`OBSERVE`

What	**EPC List**	A list containing one element: the SSCC of the pallet (instance-level identification)	`urn:epc:id:sscc:9521141.0123456789`
			or
			`https://id.gs1.org/00/095211411234567892`

When	**Event Time**	The date and time at which the pallet is shipped	`2014-03-15T10:11:12Z`
	Event Time Zone Offset	The time zone offset in effect where the pallet was shipped	`-05:00`

Where	**Read Point**	Shipping dock #2 of building 10	`urn:epc:id:sgln:9521141.11111.2`
			or
			`https://id.gs1.org/414/9521141111116/254/2`

	Business Location	(omitted)	(omitted)
Why	**Business Step**	Shipping (from CBV)	`shipping`
	Disposition	In Transit (from CBV)	`in_transit`

Dim	Data Element	Design Choice (section 4.6)	Actual EPCIS Event Contents

	Business Transaction List	A list containing two business transaction references: the Retailer's purchase order and the Manufacturer's invoice.	*Type* `po`
			`urn:epcglobal:cbv:bt:5012345678900:1234`
			Type `inv`
			`urn:epcglobal:cbv:bt:0614141111114:9876`

	Source List	A list containing one source of type "owning party," indicating the Manufacturer as the owning party at the source	*Type* `owning_party`
			as SGLN:
			`urn:epc:id:sgln:9521141.11111.0`
			or
			`https://id.gs1.org/414/9521141111116`
			as PGLN:
			`urn:epc:id:pgln:9521141.11111`
			or
			`https://id.gs1.org/417/9521141111116`

	Destination List	A list containing one source of type "owning party," indicating the Retailer as the intended owning party at the destination	*Type* `owning_party`
			as SGLN:
			`urn:epc:id:sgln:9521345.67890.0`
			or
			`https://id.gs1.org/414/9521345678903`
			as PGLN:
			`urn:epc:id:pgln:9521345.67890`
			or
			`https://id.gs1.org/417/9521345678903`

4.8 Step 8: Document the Visibility Events in a Visibility Data Matrix

You must complete Steps 5 through 7 for every one of the business steps you identified in Step 4. This sounds tedious, but typically you will find there is quite a bit of repetition and so it gets easier after the first three or four events.

When you are all done, summarise the results in a matrix that has a column for each visibility event and a row for each data element in the EPCIS data model. This looks like the tables in the previous section, extended to have a column for each event. A spreadsheet is a good tool to create this matrix.

Here's what a matrix might look like for events V1 through V4 in our example:

Table 4-7 Example Visibility Data Matrix

Dim	Data Element	V1	V2	V3	V4
	Description	Print and apply case label	Print the pallet label	Pack cases into pallet	Ship the pallet
	Event Type	Object Event	Object Event	Aggregation Event	Object Event
	Action	`ADD`	`ADD`	`ADD`	`OBSERVE`
What	**EPC List**	SGTIN of case	SSCC of pallet	Parent: SSCC of pallet Children: SGTINs of cases	SSCC of Pallet
When	**Event Time**	Current date/time	Current date/time	Current date/time	Current date/time

Dim	Data Element	V1	V2	V3	V4
	Event Time Zone Offset	Local timezone offset	Local timezone offset	Local timezone offset	Local timezone offset
Where	**Read Point**	SGLN of packaging line	SGLN of packaging line	SGLN of packaging line	SGLN of loading dock door
	Business Location	GLN of factory	GLN of factory	GLN of factory	(omitted)
Why	**Business Step**	`commissioning`	`commissioning`	`packing`	`shipping`
	Disposition	`active`	`active`	`in_progress`	`in_transit`
	Business Transaction List	(omitted)	(omitted)	(omitted)	Retailer's GLN + PO # Manufacturer's GLN + Invoice #
	Source List	(omitted)	(omitted)	(omitted)	`owning_party`: Manufacturer's GLN or PGLN
	Destination List	(omitted)	(omitted)	(omitted)	`owning_party`: Retailer's GLN or PGLN

This example matrix shows the event content described in words, as we did in Step 6. It would also be appropriate to include examples showing the specific identifier choices made in Step 7 (omitted here for reasons of space).

The next section provides some further examples of how to design EPCIS events for specific situations.

5 Advanced EPCIS Modelling

This section explores other business processes and shows how to model them using EPCIS events.

5.1 Aggregation/Disaggregation

Many business processes involve creating physical *aggregations*, where child object are packed into or onto a parent object. An aggregation has the following characteristics:

- When in a state of aggregation, the parent object and children objects may be assumed to be at the same place at the same time.
- The parent object and children objects retain their identity while in a state of aggregation. The aggregation may be reversed (disaggregated), so that the original parent and/or children objects are separate. This is in contrast to a *transformation*, in which inputs are irreversibly converted into outputs having a different identity (see section 5.5).

Examples of commonly occurring aggregations including the following:

Table 5-1 Examples of Commonly Occurring Aggregations

Description	Parent Object and its Identifier	Child Objects and their Identifiers
Items packed into a homogeneous case	Case (SGTIN)	Item (SGTIN)
Items packed into an inhomogeneous (heterogeneous) case	Case (SSCC)	Item (SGTIN)
Cases packed onto a pallet	Pallet (SSCC)	Case (SGTIN or SSCC)
Pallets loaded into a reusable shipping container	Container (GRAI)	Pallet (SSCC)
Shipping containers loaded onto a vessel, train, etc	Vessel (GIAI)	Container (GRAI)
Components installed into a chassis	Chassis (GIAI)	Component (GIAI or CPID)

The examples above all assume the child objects are identified with instance-level identification, but it is also possible to have children identified with class-level identification. The parent, however, must always be identified with an instance-level identifier.

A common reason for tracking aggregations is to allow for *inference*, in which a business application infers that all aggregated objects are present when only one is observed. For example, in the example from section 4, the EPCIS event for the shipping step only included the SSCC of the pallet, but the receiver may infer that all of the cases were shipped, too. In making this inference, the receiver is relying on (a) having the EPCIS event for the packing step, in which the aggregation is created; and (b) knowing that there are no disaggregation events between the packing step and the shipping event.

5.1.1 Aggregation and Disaggregation

The *Action* data element in an EPCIS Aggregation Event says what happened to the aggregation during the event:

Table 5-2 Action Values for Aggregation Events

Action	Meaning
`ADD`	The children were aggregated to the parent. Following the event, the children may be assumed to be physically aggregated to the parent (and therefore also to each other).
`OBSERVE`	The parent and children were observed to be in a state of aggregation, but no children were added or removed during the event. For *Action* `OBSERVE` only, the parent may be omitted, indicating that the children were observed to be in a state of aggregation but the identity of the parent could not be verified during the event.
`DELETE`	The children were disaggregated from the parent. Following the event, the children may be assumed to be physically separate from the parent and from each other. For *Action* `DELETE` only, the children may be omitted, indicating that *all* children have been disaggregated from the parent.

To illustrate, here is a business process consisting of five steps:

1. A shipper packs five homogeneous cases (each identified by an SGTIN) onto a pallet (identified by an SSCC).
2. The shipper ships the pallet, only noting the pallet identifier.
3. The receiver receives the pallet and also verifies all of the case identifiers.
4. The receiver unpacks two cases from the pallet.
5. The receiver unpacks the remaining cases from the pallet.

The following table shows the content of the five EPCIS events corresponding to these steps (the *When* and *Where* dimensions are omitted for the sake of brevity):

Table 5-3 Example EPCIS Aggregation Event Information Content

Dim	Data Element	V1	V2	V3	V4	V5
	Description	Pack cases onto pallet	Ship pallet	Receive pallet	Unpack two cases	Unpack remaining cases
	Event Type	Aggregation Event	Object Event	Aggregation Event	Aggregation Event	Aggregation Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`	`DELETE`	`DELETE`
What	**EPC List**	Parent: SSCC of pallet Children: SGTINs of 5 cases	SSCC of pallet	Parent: SSCC of pallet Children: SGTINs of 5 cases	Parent: SSCC of pallet Children: SGTINs of 2 cases	Parent: SSCC of pallet Children: (omitted)
Why	**Business Step**	`packing`	`shipping`	`receiving`	`unpacking`	`unpacking`

Dim	Data Element	V1	V2	V3	V4	V5
	Disposition	`in progress`	`in transit`	`in progress`	`in progress`	`in progress`

5.1.2 Multiple Levels of Aggregation

Some business processes may involve multiple levels of aggregation; for example, items packed into cases and those cases packed onto a pallet. In such cases, the parents of the inner aggregations are the children of the outer aggregation.

This is modelled in EPCIS, straightforwardly, by having multiple aggregation events, one for each parent at every level. For example, if five items are packed into a case, and three such cases are packed onto a pallet (for a total of 15 items), there will be a total of four aggregation events: three events that aggregate items into cases, and one that aggregates the cases onto a pallet. Here is how that would look, assuming homogeneous cases identified by SGTIN and a pallet identified by SSCC (the *When* and *Where* dimensions are omitted for the sake of brevity):

Table 5-4 Example EPCIS Aggregation Event Information Content for a Two-Level Hierarchy

Dim	Data Element	V1	V2	V3	V4
	Description	Pack items 1 – 5 into case 101	Pack items 6 – 10 into case 102	Pack items 11 – 15 into case 103	Pack cases 101, 102, and 103 onto pallet 1001
	Event Type	Aggregation Event	Aggregation Event	Aggregation Event	Aggregation Event
	Action	`ADD`	`ADD`	`ADD`	`ADD`
What	**EPC List**	Parent: SGTIN of case 101 Children: SGTINs of items 1 – 5	Parent: SGTIN of case 102 Children: SGTINs of items 6 – 10	Parent: SGTIN of case 103 Children: SGTINs of items 11 – 15	Parent: SSCC of pallet 1001 Children: SGTINs of cases 101 – 103
Why	**Business Step**	`packing`	`packing`	`packing`	`packing`
	Disposition	`in_progress`	`in_progres`	`in_progress`	`in_progress`

5.2 Drop Shipment

The *Source* and *Destination* data elements in EPCIS events provide detailed information about process steps that are part of a business transfer – the conveyance of ownership and/or possession of objects from one party to another. Each *Source* or *Destination* in an EPCIS event carries a type; the CBV defines the following three types that may be used:

Table 5-5 Source/Destination Types Defined in the CBV

CBV Source/Destination Type	Meaning
`owning_party`	The Source or Destination is an identifier for the party that relinquishes ownership (source) or receives ownership (destination) of the objects as a result of the business transfer.
`possessing_party`	The Source or Destination is an identifier for the party that relinquishes physical possession (source) or receives physical possession of the objects as a result of the business transfer.
`location`	The Source or Destination is an identifier of the physical location from where the objects are transferred (source) or to where the objects are transferred (destination). A source of type location for a shipping business step should be consistent with the read point on that event. A destination of type location for a receiving business step should likewise be consistent with the read point on that event.

In the simplest business transfer scenario, the owning party and possessing party are identical at the source and at the destination, and the location is also consistent with those parties. However, more complex scenarios may also be represented.

Consider a "drop shipment" scenario. In this scenario, a pharmaceutical manufacturer M sells product to a wholesaler W who in turn sells the product to a hospital H. Rather than physically warehousing the product, the wholesaler arranges for M to ship directly to H. The wholesaler still retains ownership of the product, however, until a subsequent sale transaction with H takes place.

The following two events show how this scenario can be expressed in EPCIS (the *When* dimension is omitted for the sake of brevity):

Table 5-6 EPCIS Event Information Content for Example "Drop Shipment" Scenario

Dim	Data Element	V1	V2
	Description	Manufacturer M drop ships to Hospital H	Shipment arrives at Hospital H
	Event Type	Object Event	Object Event
	Action	OBSERVE	OBSERVE
What	**EPC List**	SSCC of logistic unit	SSCC of logistic unit
Where	**Read Point**	GLN of M’s distribution center	GLN of H’s receiving area
	Business Location	(omitted)	GLN of H’s facility
Why	**Business Step**	shipping	arriving
	Disposition	in_transit	in_progress
	Source	*Type* owning_party GLN of M	*Type* owning_party GLN of M
	Source	*Type* possessing_party GLN of M	*Type* possessing_party GLN of M
	Source	*Type* location GLN of M’s distribution center	*Type* location GLN of M’s distribution center
	Destination	*Type* owning_party GLN of W	*Type* owning_party GLN of W
	Destination	*Type* possessing_party GLN of H	*Type* possessing_party GLN of H
	Destination	*Type* location GLN of H’s receiving area	*Type* location GLN of H’s receiving area

5.3 Class-Level Tracing

As discussed in section 4.6.1, EPCIS allows objects to be identified at the instance level or at the class level. Most of the examples in this guideline, including all of the examples preceding this section, use instance-level identification exclusively. This section describes some of the special considerations that apply when class-level identification is used.

5.3.1 Inherent Limitations of Traceability Using Class-Level Identification

Class-level identification has inherent limitations in comparison to instance-level identification. Instance-level identification makes it possible to determine precisely which EPCIS events refer to a specific object, and therefore whether two EPCIS events at different times refer to the same object. In contrast, class-level identification refers to a class of objects that cannot be differentiated from each other. The impact to class-level traceability systems is that they need to be designed to accommodate ambiguity.

Consider, for example, the following sequence of events using class-level identification:

- **V1**: Manufacturer creates 20 new product instances (each identified by GTIN and Lot only)
- **V2**: Manufacturer ships 10 product instances to a receiver

- **V3**: Manufacturer ships 10 more product instances to the same receiver
- **V4**: Receiver receives 10 product instances

The following table shows the content of these EPCIS events:

Table 5-7 Example EPCIS Event Information Content Using Class-Level Identification.

Dim	Data Element	V1	V2	V3	V4
	Description	Manufacture 20 new product instances	Ship 10 product instances	Ship 10 more product instances	Receive 10 product instances
	Event Type	Object Event	Object Event	Object Event	Object Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`	`OBSERVE`
When	**Event Time**	15 July, 10am	16 July, 10am	17 July, 10am	25 July, 10am
What	**EPC Quantity List**	GTIN X, Lot 12, 20 units	GTIN X, Lot 12, 10 units	GTIN X, Lot 12, 10 units	GTIN X, Lot 12, 10 units
Where	**Read Point**	SGLN of mfr line	SGLN of manufacturer's loading dock	SGLN of manufacturer's loading dock	SGLN of receiver's loading dock
	Business Location	GLN of manufacturer	(omitted)	(omitted)	GLN of receiver
Why	**Business Step**	`creating_class_instance`	`shipping`	`shipping`	`receiving`
	Disposition	`active`	`in_transit`	`in_transit`	`in_progress`

In this example, it is impossible to know whether the 10 units of GTIN X, Lot 12, received on 25 July in event V4 are the 10 units shipped on 16 July (event V2) or the 10 units shipped on 17 July (event V3). This is not a limitation of EPCIS; it is a fundamental limitation of using class-level identification.

A consequence of this is that common tracking and tracing tasks may be more complex using class-level data. Consider a product recall scenario, where the objective is to determine the current location of all instances of a given lot so that those instances may be removed from the supply chain. If instance-level identification is used, each instance of the lot has a unique serial number that is known from the commissioning business step. The recall application simply has to find the most recent EPCIS event for each instance identifier, and the business location of each event indicates the current location (at least to the extent inferable from EPCIS data). Each instance identifier may appear in more than one EPCIS event, but because a given instance cannot be in two places at once it is the latest event for each instance that gives its current location.

Now consider trying the same strategy with lot-level identification, in a situation where different instances of the same lot may take different paths through the supply chain. Merely finding the latest EPCIS event for that lot does not necessarily locate all of the objects. In the example above, the latest EPCIS event for Lot 12 is Event V4, but that only accounts for 10 of the 20 units. The other 10 units are in still in transit, corresponding to Event V2 or V3. A more complex analysis that attempts to tally the quantities that enter and exit each site is needed in order to identify all of the locations where the lot currently resides.

Applications using class-level identification must consider carefully how the data will be used and what limitations will naturally arise.

5.3.2 Beginning-of-Life Events for Class-Level Identification

When instance-level identification is used, any given instance will have exactly one beginning-of-life event bearing that instance identifier. Such an event is either an Object Event with Action ADD, or a Transformation Event (in which the instance identifier is an output). The business step is `commissioning` from the CBV or some more specialised business step from another vocabulary whose semantics are similar to `commissioning`.

With class-level identification, there may be many beginning-of-life events bearing the same class identifier, each such event representing the beginning of life for an additional quantity of the class.

For example, a manufacturing process may create a pallet's worth of product each hour and generate an EPCIS event for each pallet manufactured, with all the pallets in one day's production constituting a single lot. Each hourly EPCIS event represents the beginning of life for the instances produced within that hour.

The CBV defines `commissioning` for a class-level identifier to denote the process of associating an identifier *not previously used* with one or more objects within the class. In other words, `commissioning` not only represents the beginning of life for the objects, but also the beginning of life of the identifier. Only one EPCIS event with business step `commissioning` should exist for a given identifier.

To handle the case of multiple beginning-of-life events for the same class, the CBV also defines `creating_Class_instance` as an additional business step type. Unlike `commissioning`, `creating_class_instance` only implies the beginning of life of the objects, without implying anything about the life of the identifier.

In a situation where the business process is aware when a class level identifier is used for the first time (e.g., when a new lot of a product is initiated), the business step `commissioning` may be used for the first EPCIS event that creates instances of the new lot, and `creating_class_instance` for any subsequent events that create additional instances of that lot. Sometimes, it may not be feasible or possible to know which EPCIS event is the first use of a class-level identifier; in those cases, `creating_class_instance` may be used for all events that create instances of the class.

5.3.3 Class-Level Identification In Aggregation

An Aggregation Event may have children that are identified using class-level identification. The parent, however, must always be identified using instance-level identification.

For example, supposed that homogeneous cases of product are picked to order, shipped, and received, where the cases are only identified with GTIN and Lot. The events might look like this (the *When* and *Where* dimensions are omitted for the sake of brevity):

Table 5-8 EPCIS Event Information Content for Aggregation of Children Identified at Class Level

Dim	Data Element	V1	V2	V3
	Description	Pack cases onto pallet	Ship pallet	Receive pallet
	Event Type	Aggregation Event	Object Event	Object Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`
What	**EPC List**	Parent: SSCC of pallet Children: GTIN X, Lot 12, 10 units GTIN Y, Lot 52, 20 units	SSCC of pallet	SSCC of pallet
Why	**Business Step**	`packing`	`shipping`	`receiving`
	Disposition	`in_progress` (CBV)	`In Transit`	`In Progress`

In this example, the receiver can use the prior aggregation event to infer that the pallet it receives contains 10 units of GTIN X (Lot 12) and 20 units of GTIN Y (Lot 52). Subsequent events might disaggregate product from the pallet, again identifying the specific quantities of the classes that are disaggregated.

It is not permitted to use a class-level identifier to identify the parent of an aggregation. The reason is that inference is only possible if each aggregation has a distinct identity (as represented by the parent identifier), and if inference is not possible then attempting to record the aggregation is of no value.

5.3.4 Mixing Instance-Level and Class-Level Identification in the Same Event

It is possible for one EPCIS event to include a mix of both instance-level and class-level identification. For example, a pallet picked to order may include one product identified by SGTIN, another by GTIN+Lot, and a third identified only by GTIN. Here is an example (the *When* and *Where* dimensions are omitted for the sake of brevity):

Table 5-9 EPCIS Aggregation Event Information Content with Children Identified at Both Instance and Class Level

Dim	Data Element	V1	V2	V3
	Description	Pack cases onto pallet	Ship pallet	Receive pallet
	Event Type	Aggregation Event	Object Event	Object Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`
What	**EPC List**	Parent: SSCC of pallet Children: GTIN X, Serial 101 GTIN X, Serial 102 GTIN X, Serial 103 GTIN Y, Lot 12, 10 units GTIN Z, 20 units	SSCC of pallet	SSCC of pallet
Why	**Business Step**	`packing`	`shipping`	`receiving`
	Disposition	`in_progress`	`in_transit`	`in_progress`

As before, the aggregation event allows the receiver to infer the contents of the pallet, which in this case uses a mix of instance-level and class-level identification.

Mixing of instance-level and class-level identification is particularly common in transformation events arising in manufacturing, where the ingredients in a manufacturing process include "primary" ingredients that are identified at the instance-level and "secondary" ingredients identified only at the class level. For example:

- Inputs:
  - Tuna loin (each loin is individually serialised and identified by SGTIN – instance-level)
  - Olive oil (identified by GTIN+Lot – class level)
  - Empty can (identified by GTIN, in order to distinguish two possible suppliers of cans)
- Outputs:
  - Canned tuna (each can identified either at the instance level (SGTIN) or the class level (GTIN+Lot), depending on the business requirement)

It is important to note that when instance-level and class-level identifiers are mixed in the same EPCIS event, each identifier is understood to refer to a *different* object.

If the desire is to indicate the Lot number associated with items identified by GTIN+Serial, only the SGTINs should be included in the event, and the Lot number provided via instance/lot master data on the commissioning event for those serial numbers (see section 5.4).

5.4 Instance/Lot Master Data (ILMD)

As explained in section 7.3.6 of the EPCIS 1.1 standard, Instance/Lot Master Data (ILMD) is data that describes a specific instance of a physical or digital object, or a specific batch/lot of objects that are produced in batches/lots. It is similar to ordinary master data, which also consists of a set of descriptive attributes that provide information about objects. But whereas master data attributes have the same values for a large class of objects, (e.g., for all objects having a given GTIN), the values of ILMD attributes may be different for much smaller groupings of objects (e.g., a single batch or lot), and may be different for each object (i.e., different for each instance).

Conceivably, instance and lot level master data could be communicated between trading partners outside of EPCIS, just as GTIN-level master data may be communicated outside EPCIS using the Global Data Synchronisation Network (GDSN). However, at this time there are no well-established mechanisms for communication of instance or lot level master data. For this reason, EPCIS provides a means to attach instance and lot level master data to the EPCIS event that marks the beginning of life for a new instance.

In the case of objects identified at the instance level, master data for the instance is carried in the commissioning event for that instance, or in a transformation event if the instance is created as the output of a transformation. For example, the following table shows the content of three EPCIS events, including instance-level master data at the commissioning step (the "when" dimension is omitted for brevity):

Table 5-10 EPCIS Event Information Content Showing Instance/Lot Master Data (ILMD)

Dim	Data Element	V1	V2	V3
	Description	Manufacture new product instance	Ship product	Receive product
	Event Type	Object Event	Object Event	Object Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`
What	**EPC List**	SGTIN of product instance	SGTIN of product instance	SGTIN of product instance
Where	**Read Point**	SGLN of manufacturing line	SGLN of manufacturer’s loading dock	SGLN of receiver’s loading dock
	Business Location	GLN of manufacturer	(omitted)	GLN of receiver
Why	**Business Step**	`commissioning`	`shipping`	`receiving`
	Disposition	`active`	`in_transit`	`in_progress`
	ILMD: Expiry	Expiration date of product instance		
	ILMD: Lot	Lot number of product instance		

Note that when an object is identified at the instance level, its lot number (if any) is a master data attribute of that instance.

In the example above, if the receiver wishes to obtain the master data for the product instance it receives, it queries the manufacturer for the event having the specified SGTIN in the *What* dimension and having business step `commissioning` (from the CBV).

In the XML representation of an EPCIS event, ILMD is expressed using elements defined in XML namespaces other than the EPCIS namespace. The CBV standard defines commonly used master data attributes, using the XML namespace `urn:epcglobal:cbv:mda`. Those master data attributes have definitions that match definitions used in other GS1 standards including GDSN and GS1 EDI. Other master data attributes may be defined in other standards or otherwise agreed to in advance by trading partners; such attributes must have an XML namespace other than the EPCIS or CBV namespaces.

Here is how the V1 event above might look in **XML**, using **EPC URNs** for identification in the *what* and *where* dimensions:

```
<epcis:EPCISDocument
    xmlns:epcis="urn:epcglobal:epcis:xsd:1"
    xmlns:cbvmda="urn:epcglobal:cbv:mda"
    schemaVersion="1.2"
    creationDate="2014-05-30T15:14:27.000-04:00">
  <EPCISBody>
    <EventList>
      <ObjectEvent>
      <eventTime>2023-02-02T23:08:00.11+01:00</eventTime>
      <eventTimeZoneOffset>+01:00</eventTimeZoneOffset>
       <epcList>
          <epc>urn:epc:id:sgtin:9521141.012345.400</epc>
        </epcList>
        <action>ADD</action>
        <bizStep>urn:epcglobal:cbv:bizstep:commissioning</bizStep>
        <disposition>urn:epcglobal:cbv:disp:active</disposition>
       <readPoint>
           <id>urn:epc:id:sgln:9521141.54321.0</id>
       </readPoint>
       <bizLocation>
           <id>urn:epc:id:sgln:9521141.54377.0</id>
       </bizLocation>
        <extension>
          <ilmd>
            <cbvmda:itemExpirationDate>2024-03-15
                 </cbvmda:itemExpirationDate>
            <cbvmda:lot>A123</cbvmda:lot>
          </ilmd>
        </extension>
      </ObjectEvent>
    </EventList>
  </EPCISBody>
</epcis:EPCISDocument>
```

Here is how the V1 event above might look in **JSON-LD**, using **GS1 Digital Link URIs** for identification in the *what* and *where* dimensions:

```
{
  "@context": [
    "https://ref.gs1.org/standards/epcis/2.0.0/epcis-context.jsonld"
  ],
  "type": "EPCISDocument",
  "schemaVersion": "2.0",
  "creationDate": "2014-05-30T15:14:27.000-04:00",
  "epcisBody": {
    "eventList": [
      {
        "type": "ObjectEvent",
        "eventTime": "2023-02-02T23:08:00.11+01:00",
        "eventTimeZoneOffset": "+01:00",
        "epcList": [
          " https://id.gs1.org/01/09521141123454/21/400"
        ],
        "action": "ADD",
        "bizStep": "commissioning",
        "disposition": "active",
        "readPoint": {
          "id": "https://id.gs1.org/414/9521141543214"
        },
        "bizLocation": {
          "id": "https://id.gs1.org/414/9521141543771"
        },
        "ilmd": {
          "cbvmda:itemExpirationDate": "2024-03-15",
          "cbvmda:lot": "A123"
        }
      }
    ]
  }
}

```

Lot-level master data works the same as instance-level master data. In contrast to instance-level identification, when lot-level identification is used there may be many beginning-of-life events (object or transformation events having business step commissioning or creating-class-instance) for the same lot. The ILMD for that lot may be included in *all* such beginning-of-life events, with the proviso that the content of the ILMD must be identical for all events pertaining to the same lot. When both commissioning and creating-class-instance business steps are used, it is acceptable to include ILMD only with the commissioning steps.

5.5 Transformation

The EPCIS Transformation Event is used to represent a business process step in which one or more objects are fully or partially consumed as inputs and one or more objects are produced as outputs. The Transformation Event captures the relationship between the inputs and the outputs, namely that any of the inputs may have contributed in some way to each of the outputs.

In contrast to aggregation, a transformation is irreversible. Following the transformation, the inputs that were consumed no longer exist, and the outputs are brand-new objects that did not exist prior to the transformation. In this way, a transformation event functions as the beginning-of-life event for the outputs and as end-of-life for the inputs (unless the inputs are not fully consumed).

Examples of commonly occurring transformations including the following:

Table 5-11 Examples of Transformation Business Processes

Description	Input Objects and their Identifiers	Output Objects and their Identifiers
Raw materials combined into a mixture	Raw materials (a separate SGTIN, GTIN+Lot, or GTIN for each raw material)	Mixed product (SGTIN, GTIN+Lot, or GTIN for each packaging variation)
Primal cuts of meat combined, divided, and packaged into packaged meat products	Primal meat cuts (SGTIN), seasonings or other secondary ingredients (GTIN+Lot), and sterile packaging material (GTIN)	Packaged meat product (SGTIN or GTIN+Lot)
Bulk pharmaceutical product repackaged into smaller saleable units	Bulk pharmaceutical (SGTIN or GTIN+Lot)	Saleable pharmaceutical units (SGTIN or GTIN+Lot)

A common reason for tracking transformations is to give business processes an understanding of what inputs might have affected what outputs. For example, if a primal cut of meat coming from a specific ranch is discovered to have bacterial contamination, the transformation event allows this to be traced forward to identify all of the finished meat products that might be affected by the contaminated primal cut. Conversely, if a finished product is discovered to be contaminated, the transformation allows this to be tracked backward to identify all of the ingredients, which then may be traced forward to find additional finished product that might be affected.

5.5.1 Transformation Event Example

Consider the following manufacturing process:

- Inputs:
  - Tuna loin (each loin is individually serialised and identified by GTIN X plus serial – instance-level)
  - Olive oil (identified by GTIN Y + Lot – class level)
  - Empty can (identified by GTIN Z, in order to distinguish two possible suppliers of cans)
- Outputs:
  - Canned tuna (identified by GTIN Q + Lot – class level)

Here is a transformation event for one run of this process (the *When* and *Where* dimensions are omitted for the sake of brevity):

Table 5-12 Example EPCIS Transformation Event Information Content

Dim	Data Element	V1
	Description	Manufacture canned tuna from raw ingredients
	Event Type	Transformation Event
What	**EPC List**	Inputs: GTIN X, Serial 10 GTIN X, Serial 45 GTIN X, Serial 97 GTIN Y, Lot 12, 10 litres GTIN Z, 100 units Outputs: GTIN Q, Lot 999, 100 units
Why	**Business Step**	`creating_class_instance`

Dim	Data Element	V1
	Disposition	`active`

As the transformation is the beginning of life for the outputs, a beginning-of-life business step and disposition are used. In this case, the transformation creates new instances of Lot 999, so `Creating Class Instance` is used as the business step. If it is known that this event creates Lot 999 for the first time, `Commissioning` could be used instead.

5.5.2 Long-Running Transformations

Sometimes a transformation runs over a long period of time, in which inputs are added periodically and outputs extracted periodically. For example, a mixing process might consume inputs in several batches over the course of a product run, with outputs withdrawn even as more inputs are added.

A long-running transformation can be modelled with a single EPCIS event that lists all of the inputs and all of the outputs that were involved over the entire interval of time. This raises a question about what event time is appropriate; most often, the time at which the process completed is the appropriate event time to use.

It is not always desirable, however, to model a long-running transformation as a single EPCIS event. This is especially true if some of the output objects are subject to further business steps even before the transformation has completed. For example, consider a process that mixes input ingredients to create cans of paint, where a production run involving the same mixing vat runs continuously for a week. The process may produce cans of paint on Monday, and those cans are shipped on Tuesday, even though more cans from the same vat are extracted on Wednesday and Thursday, with the entire transformation completing on Friday. In this situation, it may be necessary to have an EPCIS event to represent Monday’s production so that the new identifiers for the cans of paint are available to be used in a shipping event generated on Tuesday.

To model such situations, a transformation event may be split into multiple EPCIS events. To maintain the relationship between all inputs and outputs, the multiple transformation events are linked by using a transformation identifier. This is simply a unique identifier that is the same for all EPCIS events belonging to the same transformation (i.e., where there is a relationship between inputs and outputs), and different from the transformation identifier used in other, unrelated events.

The following set of events is equivalent to the transformation of the previous section, plus an added event showing the shipment of the first few cans of tuna to be produced.

Table 5-13 Example EPCIS Transformation Event Information Content Linked Via Transformation ID

Dim	Data Element	V1	V2	V3	V4
	Description	Add first set of ingredients to new batch	Withdraw first set of cans	Ship first set of cans	Add remaining ingredients and finish manufacturing
	Event Type	Transformation Event	Transformation Event	Object Event	Transformation Event
What	**Transformation ID**	Xform 123	Xform 123		Xform 123
	EPC List	Inputs: GTIN X, Serial 10 GTIN X, Serial 45 GTIN Y, Lot 12, 5 litres GTIN Z, 40 units Outputs: (omitted)	Inputs: (omitted) Outputs: GTIN Q, Lot 999, 30 units	GTIN Q, Lot 999, 30 units	Inputs: GTIN X, Serial 97 GTIN Y, Lot 12, 5 litres GTIN Z, 60 units Outputs: GTIN Q, Lot 999, 70 units
Why	**Business Step**	`creating_class_instance`	`creating_class_instance`	`shipping`	`creating_class_instance`
	Disposition	`active`	`active`	`in_transit`	`active`

The CBV provides templates that may be used to construct transformation IDs.

5.6 Coupons and Vouchers

EPCIS is not limited to tracking physical objects. EPCIS may also be used to track digital objects such as digital trade items (music downloads, electronic books, etc.), digital documents (electronic coupons, etc.), and so forth. In most cases, the business processes for digital objects are similar to the processes for physical objects, and the same CBV business steps and dispositions can be used.

This section illustrates EPCIS applied to digital objects by showing two processes for tracking the lifecycle of a digital coupon. A digital coupon is an offer from a manufacturer or retailer to provide something of value (a cash rebate, a discount, or an additional trade item) to a consumer when the consumer purchases a particular trade item. A particular offer from a manufacturer or retailer may be identified by a Global Coupon Number (GCN), and a particular instance of that offer as issued to and redeemed by a consumer may be identified by a Global Coupon Number with a serial number (SGCN). Master data associated with the GCN may provide details about the offer, such as the GTIN of the required purchase, the amount of the cash rebate, etc. A digital voucher, such as a voucher issued to a consumer by a bottle recycling machine and redeemed by the consumer at point-of-sale, works in a similar manner.

Two processes are illustrated here: a simple process where a coupon is issued by a manufacturer and redeemed by a retailer at point of sale, and a more complex process involving a coupon broker.

Both of these examples are intended to illustrate the general concept of using EPCIS for digital objects. For specific details on how to model coupon processes, see GS1 application standards or local recommendations for this purpose.

5.6.1 Simple Coupon Process

In the simplest coupon process, there are just two steps that require EPCIS events:

- **V1**: A customer is issued a digital coupon by a coupon issuer. Typically the coupon issuer is a retailer, but it could also be a manufacturer or a third party. The coupon is often issued to the customer via a mobile application that the customer uses on his device. The coupon's SGCN is stored with that application for use in the next step. An EPCIS event is generated to indicate that the coupon is now active.
- **V2**: The customer redeems the coupon at a point-of-sale terminal during checkout at a retail store (whether brick-and-mortar or online). The point-of-sale application verifies that the coupon is valid and that the conditions of the offer are met; if so, the coupon is redeemed and an EPCIS event generated to indicate that the coupon is no longer active.

These two events are indicated in EPCIS using a business step of commissioning and decommissioning, respectively.

Table 5-14 Example EPCIS Event Information Content for Simple Digital Coupon Business Process

Dim	Data Element	V1	V2
	Description	Issue a digital coupon	Redeem a digital coupon
	Event Type	Object Event	Object Event
	Action	`ADD`	`DELETE`
When	**Event Time**	15 July, 10am	16 July, 10am
What	**EPC**	SGCN X	SGCN X
Where	**Read Point**	SGLN of coupon issuer (typically a party GLN if there is no physical location involved, but could be SGLN of a physical location such as a kiosk where the coupon is dispensed)	SGLN of retailer point-of-sale terminal (or a party GLN if there is no physical location involved, as in an online sale)
	Business Location	(omitted)	(omitted)

Dim	Data Element	V1	V2
Why	**Business Step**	`commissioning`	`decommissioning`
	Disposition	`active`	`inactive`
	ILMD	(see below)	(none)

In the commissioning step (V1), ILMD could be used to record attributes of the coupon such as the associated GTIN, the date range during which the coupon is redeemable, the customer's loyalty card number, and so on.

Once EPCIS events are captured, EPCIS queries could be used to determine the total number of coupons activated in a given timeframe, the total number of coupons for a given GCN (class level), all SGCNs not yet redeemed (but still valid), the number of redemptions and the time period between coupon activation/redemption, and so on.

5.6.2 Coupon Example With Coupon Broker

The example in the previous section assumes that only two events require tracking in EPCIS: the issuance of the coupon to a customer, and the redemption of the coupon at point of sale. But as in any business process, the process of coupon redemption may be more complex in practice, and it may be useful to model more steps of the process in EPCIS. For example, in a more complex scenario there might be four steps that could be recorded using EPCIS:

- **V1**: A coupon issuer (retailer or manufacturer) issues a block of coupons to a coupon distributor (e.g., an Internet application provider specialising in electronic coupons).
- **V2**: A customer is issued a digital coupon by the coupon distributor.
- **V3**: The customer redeems the coupon at a point-of-sale terminal during checkout at a retail store (whether brick-and-mortar or online).
- **V4**: Final settlement takes place between the coupon distributor and the issuer.

As in the previous example, each of these business steps could be modelled as an EPCIS event. In this example, commissioning takes place in V1 when the coupon is issued to the distributor, not when the distributor issues the coupon to the consumer (the latter being more akin to a receiving operation by the consumer). And decommissioning takes place during the final settlement step V4, not when the consumer redeems the coupon.

As the CBV does not include all of the business step and disposition values that would be needed to model all four of these events, they are not illustrated here. Specific standards or guidelines for coupon tracking may be developed by GS1 in the future.

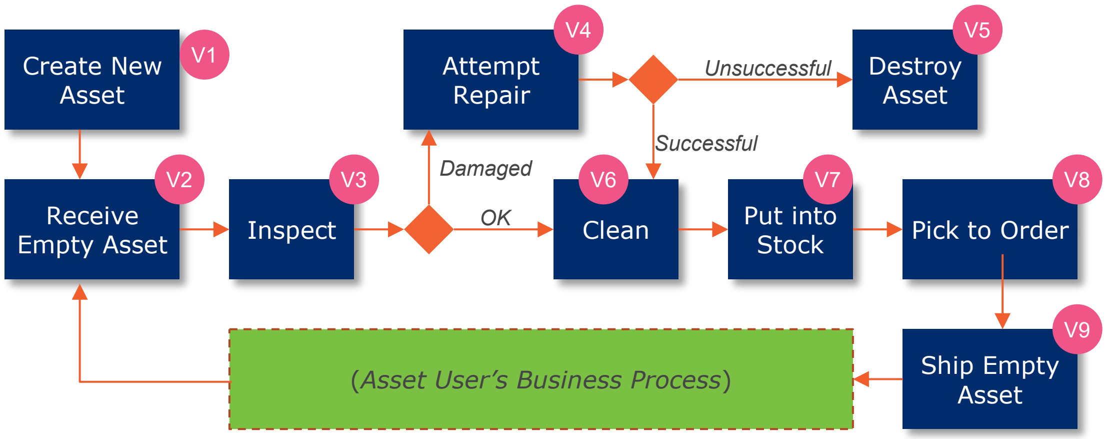
The main point here is that whether the "what" is a physical object or a digital object, the same analysis and design procedure serves to model a business process using EPCIS.

5.7 Returnable Asset Management Using GRAI

This section illustrates how EPCIS may be used to capture tracking events for returnable assets, such as pooled pallets or totes. Each returnable asset is identified using a GRAI that includes a unique serial number.

There are two interlocking business processes, one carried out by the party that manages the returnable assets, and one carried out by the users of the assets. The first process is illustrated below.

Table 5-15 Process Flowchart for Example Returnable Asset Management Business Process



This process normally runs in a cycle from V2 through V9, through the asset user's business process, and back to V2 again. Assets received from users are inspected at V3. If damaged, a repair is attempted at V4 and the asset destroyed at V5 if the asset is unrepairable. Otherwise, or if inspection showed no damage, the asset is cleaned at V6 and placed into stock with other similar assets at V7. When a user places an order for one or more empty assets, they are picked at V8, and once the user is invoiced they are shipped to the user at V9. There they enter the asset user's business process.

The party that manages the returnable assets may be interested in tracking all nine of the events illustrated above. Here is how they may be modelled in EPCIS. In each case, the *What* dimension includes the GRAI(s) of the asset(s) involved in that step and the *Where* dimension contains the appropriate location within the asset manager's facility. Below, the *What*, *When*, and *Where* dimensions are omitted for the sake of brevity.

Table 5-16 EPCIS Event Information Content for Returnable Asset Management Business Process (V1 – V4)

Dim	Data Element	V1	V2	V3	V4
	Description	Create new asset	Receive empty asset	Inspect asset	Attempt repair
	Event Type	Object Event	Object Event	Object Event	Object Event
	Action	`ADD`	`OBSERVE`	`OBSERVE`	`OBSERVE`
Why	**Business Step**	`commissioning`	`receiving`	`inspecting`	`repairing`
	Disposition	`active`	`in_progress`	(see below)	(see below)

Table 5-17 EPCIS Event Information Content for Returnable Asset Management Business Process (V5 – V9)

Dim	Data Element	V5	V6	V7	V8	V9
	Description	Destroy unrepairable asset	Clean asset	Put into stock	Pick to order	Ship empty asset
	Event Type	Object Event	Object Event	Object Event	Object Event	Object Event
	Action	`DELETE`	`OBSERVE`	`OBSERVE`	`OBSERVE`	`OBSERVE`
Why	**Business Step**	`destroying`	`cleaning` (see below)	`stocking`	`picking`	`shipping`
	Disposition	`destroyed`	`in_progress`	`in_progress`	`in_progress`	`in_transit`

Notes on these events:

- In V3, the disposition and what happens next depends on the result of the inspection:

  - If the condition of the asset is acceptable, the disposition is `in_progress` (from the CBV) and the next step is V6 (cleaning).
  - If the condition of the asset is unacceptable, the disposition is `damaged` (from the CBV) and the next step is V4 (repairing).
- In V3, the disposition and what happens next depends on the result of the repair attempt:
  - If the asset was successfully repaired, the disposition is `in_progress` (from the CBV) and the next step is V6 (cleaning).
  - If the asset could not be repaired, the disposition is `damaged` (from the CBV) and the next step is V5 (destroying).
- In V6, there is no CBV business step corresponding to "cleaning," so this is a situation where a private vocabulary element might be used.

After V9, the empty asset is used by an asset user to move goods through the user's own supply chain. There are two possibilities for how the returnable asset is used:

- The user may not be aware of or make any use of the GRAI of the asset at all. In that case, the asset is just a "dumb" pallet or tote, and its GRAI does not enter into any EPCIS events. The user may track products loaded onto the asset using their GTINs, and/or associate an SSCC with the complete logistic load carried by the asset, but either way such use is wholly unrelated to the use of the GRAI by the asset manager.
- The user may take advantage of the asset's GRAI and the bar code or RFID data carrier containing it.

5.8 User Extensions

5.8.1 User extensions in XML

The EPCIS data model is designed to include all of the relevant *What*, *When*, *Where*, and *Why* information a business application needs to understand what happened in a business process step. However, sometimes a business application has information needs that go beyond the data elements defined in the EPCIS standard. To accommodate such situations, EPCIS events may carry user/vendor extension data elements.

A user/vendor extension data element is simply any data element added to an EPCIS event. Most commonly these express additional business context and so can be considered an addition to the *Why* dimension of an event, but as there are no restrictions on the content of an extension it could pertain to any other dimension as well.

In the XML representation of an EPCIS event, an extension data element is expressed as an XML element whose XML namespaces is something other than the EPCIS namespace. Neither the EPCIS standard nor the CBV standard define any specific extension data elements, so these must either be defined in other standards (e.g., a sector-specific standard or standard promulgated within a trading group) or otherwise agreed to in advance by trading partners.

To illustrate, here is an example of an EPCIS event that has an additional data element to record the badge number of a stakeholder inspecting the objects, as might be appropriate for an inspection made while an item is in transit:

Table 5-18 EPCIS Event information content illustrating user/vendor extensions

Dim	Data Element	V1
	Description	Inspection of objects
	Event Type	Object Event
	Action	`OBSERVE`
When	**Event Time**	15 January 2023, 10am EST

Dim	Data Element	V1
What	**EPC Quantity List**	GTIN X, Serial 101 GTIN X, Serial 102 GTIN X, Serial 103
Where	**Read Point**	Geolocation: (41°40’21”N 86°15’19”W)
	Business Location	(omitted)
Why	**Business Step**	`inspecting`
	Disposition	`in_progress`
	Extension: inspector_badge_nr	`244301128`

Here is the **XML** representation of the above event:

```
<epcis:EPCISDocument xmlns:epcis="urn:epcglobal:epcis:xsd:1"
xmlns:myvoc="http://epcis.example.org/myvoc" schemaVersion="1.2"
creationDate="2014-05-30T15:14:27.000-04:00">-
 <EPCISBody>
  <EventList>
    <ObjectEvent>
      <eventTime>2023-01-15T10:00:00.000-05:00</eventTime>
      <eventTimeZoneOffset>-05:00</eventTimeZoneOffset>
      <epcList>
         <epc>urn:epc:id:sgtin:9521141.012345.101</epc>
        <epc>urn:epc:id:sgtin:9521141.012345.102</epc>
        <epc>urn:epc:id:sgtin:9521141.012345.103</epc>
      </epcList>
      <action>OBSERVE</action>
      <bizStep>urn:epcglobal:cbv:bizstep:inspecting</bizStep>
      <disposition>urn:epcglobal:cbv:disp:in_progress</disposition>
      <readPoint>
         <id>geo:41.6725,-86.255278</id>
      </readPoint>
      <myvoc:inspector_badge_nr>244301128</myvoc:inspector_badge_nr>
   </ObjectEvent>
  </EventList>
 </EPCISBody>

</epcis:EPCISDocument>
```

In the XML example above, the two extension elements are in an XML namespace defined by the Example Corporation. The use of the XML namespace not only distinguishes extensions from standard EPCIS data elements, but also ensures that Example Corporation’s extensions will not be confused with extensions of other organisations that may use the same element names.

The following guidelines should be observed in using extension elements:

- Extension elements must be agreed in advance by trading partners, otherwise they may not be correctly interpreted.
- EPCIS standard data elements should always be used in preference to extension data elements, as they will be more interoperable.
- The XML namespace URI used for the extensions should be a URI that is under the control of the organisation defining the extensions. Using an HTTP URI based on the Internet domain name of the defining organisation is a recommended approach.

- Extension elements should provide information that provides additional data about the event in which they are included. They should not be used to communicate data not related to the event. In particular, data that is properly considered as master data pertaining to an instance-level or lot-level identifier should be carried in the ILMD section of an event, not as an extension element. See section 5.4.
- An extension data element can contain any well-formed XML content, including sub-elements and attributes. However, the EPCIS SimpleEventQuery is only capable of querying extension elements whose values are numbers or strings.
- The XML element `<extension>` defined in the EPCIS XML schema should never be used to carry user or vendor extensions. The `<extension>` element is reserved for use by the EPCIS standard itself, to introduce new data elements in later versions of the EPCIS standard.
- Applications receiving EPCIS data must not reject an EPCIS event merely because it contains an extension that the application does not recognise. Such extensions should be ignored, or perhaps noted or saved without further interpretation. On the other hand, an extension whose content violates validation criteria established in advance by trading partners may be rejected on that basis.

5.8.2 User extensions in JSON-LD

EPCIS 2.0 provides support for the use of JSON and JSON-LD (JSON for Linked Data), as an alternative to XML.

JSON is simpler than XML and lacks some of the features. Two major features that are missing are explicit data types and support for multiple namespaces.

In XML and more specifically XML Schema (XSD), it's possible to state that a string such as "2022-12-20" is not just a string but should be interpreted as a string representation of a particular data type, such as xsd:date. XML also uses namespace prefixes and namespace declarations so that XML elements and attributes from multiple different namespaces can coexist within each XML document.

JSON-LD (JSON for Linked Data) is an enhanced version of JSON that adds such missing features, as well as some other features that are useful for bridging the gap with Linked Data.

JSON-LD introduces several extra keywords prefixed with the '@' symbol, such as @id, @type, @value, @language, @context.

EPCIS 2.0 hides most of these within a JSON-LD context file/resource. This means that JSON/JSON-LD can be treated as if it is just JSON data, while those who want to make full use of the Linked Data features can additionally process the JSON-LD context file/resource to obtain Linked Data.

A **JSON-LD context is used to declare a namespace prefix**. For example, that "gs1" means "https://gs1.org/voc/" (the GS1 Web vocabulary). This allows the example to use Compact URI Expressions (CURIEs) (defined in https://www.w3.org/TR/curie/) so that instead of writing each Linked Data URI in full, such as https://gs1.org/voc/gtin we can instead use a corresponding CURIE, such as gs1:gtin – and the namespace declaration takes care of that expansion.

Here is the **JSON-LD** representation of the event expressed in section 5.8.1 in XML, using **GS1 Digital Link URIs** instead of EPC URNs:

```
{
  "@context": [
    "https://ref.gs1.org/standards/epcis/2.0.0/epcis-context.jsonld",
    {
      "myvoc": "http://epcis.example.com/myvoc/"
    }
  ],
  "type": "EPCISDocument",
  "schemaVersion": "2.0",
  "creationDate": "2014-05-30T15:14:27.000-04:00",
  "epcisBody": {
    "eventList": [
      {
        "type": "ObjectEvent",
        "eventTime": "2023-01-15T10:00:00-05:00",
        "eventTimeZoneOffset": "-05:00",
        "epcList": [
          "https://id.example.org/01/09521141123454/21/101",
          "https://id.example.org/01/09521141123454/21/102",
          "https://id.example.org/01/09521141123454/21/103"
        ],
        "action": "OBSERVE",
        "bizStep": "inspecting",
        "disposition": "in_progress",
        "readPoint": {
          "id": "geo:41.6725,-86.255278"
        },
        "myvoc:inspector_badge_nr": "244301128"
      }
    ]
```

5.9 Erroneous events

As explained throughout this guideline, in EPCIS a business process is modelled by breaking it down into a series of steps, and modelling each as an EPCIS event. The net effect is that the collection of all events pertaining to a specific object (a "trace") should correctly indicate the history and current state of that object, by interpreting the events according to the semantics specified in the EPCIS and CBV standards, and any other relevant vocabulary standards.

Sometimes, it is discovered that an event recorded earlier does not accurately reflect what happened in the real world. However, neither the EPCIS Capture Interface nor the EPCIS Query Interface provides a means by which an application can delete or modify an EPCIS Event. The only way to "retract" or "correct" an EPCIS Event is to generate a subsequent event whose business meaning is to rescind or amend the effect of a prior event. The net effect is that the complete trace (including the new events and all prior events including the incorrect event) accurately reflects the history and current state, as stated in the above principle.

The preferred way to arrive at the additional events is to recognise that the discovery of an erroneous event and its remediation is itself a business process which can be modelled by creating suitable EPCIS events. In most situations, this is done using the same methods discussed in section 4.

Example 1: Company X records an EPCIS event asserting that serial numbers 101, 102, and 103 of some product were shipped to Company Y. Company Y receives the shipment and finds serial number 104 in addition to serial numbers 101, 102, 103. In discussion with Company X, it is agreed that serial 104 was indeed shipped and that the shipping event was in error. Remediation: Company X records a new EPCIS event asserting that serial number 104 was shipped, with similar contextual information as the original event.

Example 2: Company X records an EPCIS event asserting that serial numbers 101, 102, and 103 of some product were shipped to Company Y. Company Y receives the shipment and finds only serial numbers 101, 102. In discussion with Company X, it is agreed that serial 103 was not shipped but remains in Company X's inventory. They agree to reverse the billing for the third product. Remediation: Company X records a new EPCIS event asserting that the shipment of serial 103 is voided.

In the first example, the additional event uses the same business vocabulary as the first. In the second example, vocabulary specifically associated with the process of voiding a shipment is used, but it is still "ordinary" EPCIS semantics in the sense that it models the completion of a well-defined business process step. This reflects the reality that the act remediation is itself a business process, and so may be modelled as an EPCIS event.

In some situations, it either is not possible (or is highly undesirable) to remediate the history of an object by creating a new EPCIS event with ordinary semantics.

Example 3: Company X records an EPCIS event to assert that serial number 101 of product X was destroyed. This event is an Object Event with action = DELETE. Later it is discovered that serial 101 is still in storage, not destroyed. An ordinary event cannot be used to amend the history, because the semantics of action DELETE for an Object Event specify that "the objects … should not appear in subsequent events."

Example 4: Company X records an EPCIS event asserting that several products have been shipped, indicating Purchase Order 123 as a business transaction in the "why" dimension. Company Y receives the products and records a receiving event. Only then it is discovered that the purchase order reference in the shipping event is wrong: it says PO 456 instead of 123. This could be remediated using ordinary EPCIS events by Company X recording a "void shipping" event followed by a "shipping" event with the correct PO #. But this is rather undesirable from the perspective of the overall trace, especially given that there is already a receiving event.

To accommodate such situations, EPCIS includes a mechanism to construct an event whose semantics assert that the assertions made by a prior event are in error. Such an event is termed an "error declaration event."

The following sections illustrate the various approaches to correcting errors in more detail.

5.9.1 Example 1: Correction using an ordinary event – simple addition

In this example, Company X records an EPCIS event asserting that serial numbers 101, 102, and 103 of some product were shipped to Company Y. Company Y receives the shipment and finds serial number 104 in addition to serial numbers 101, 102, 103. In discussion with Company X, it is agreed that serial 104 was indeed shipped and that the shipping event was in error.

The remediation is that Company X records a new EPCIS event asserting that serial number 104 was shipped, with similar contextual information as the original event.

Both events together look like this:

Table 5-19 Example of correcting an error by adding an ordinary event with a corrective business step.

Dim	Data Element	V1	V2
	Description	Ship 3 product instances, not realising that physical shipment includes a fourth instance	Additional event recognising that the fourth instance was shipped, too
	Event Type	Object Event	Object Event
	Action	OBSERVE	OBSERVE
When	**Event Time**	15 July, 10am	15 July, 10am
What	**EPC List**	GTIN X, Serial 101, 102, 103	GTIN X, Serial 104
Where	**Read Point**	SGLN of manufacturer's loading dock	SGLN of manufacturer's loading dock
	Business Location	(omitted)	(omitted)
Why	**Business Step**	sshipping	shipping
	Disposition	in_transit	in_transit
	Source	owning_party: GLN of Company X	owning_party: GLN of Company X
	Destination	owning_party: GLN of Company Y	owning_party: GLN of Company Y

5.9.2 Example 2: Correction using an ordinary event – corrective business step

In this example, Company X records an EPCIS event asserting that serial numbers 101, 102, and 103 of some product were shipped to Company Y. Company Y receives the shipment and finds only serial numbers 101, 102. In discussion with Company X, it is agreed that serial number 103 was not shipped but remains in Company X's inventory. They agree to reverse the billing for the third product.

The remediation is that Company X records a new EPCIS event asserting that the shipment of serial 103 is voided. This uses a business step `void_shipping` which is defined specifically for this purpose. As the new event only refers to serial number 103, it does not affect the shipping event for the other serial numbers 101 and 102, so processing of those serial numbers can continue even before the `void_shipping` event is received.

Both events together look like this:

Table 5-20 Example of correcting an error by adding an ordinary event.

Dim	Data Element	V1	V2
	Description	Ship 3 product instances, not realising that physical shipment is missing one instance	Additional event to indicate that the third instance was not actually shipped
	Event Type	Object Event	Object Event
	Action	`OBSERVE`	`OBSERVE`
When	**Event Time**	15 July, 10am	18 July, 2pm
What	**EPC List**	GTIN X, Serial 101, 102, 103	GTIN X, Serial 103
Where	**Read Point**	SGLN of manufacturer’s loading dock	SGLN of manufacturer’s loading dock
	Business Location	(omitted)	SGLN of manufacturer’s warehouse
Why	**Business Step**	`shipping`	`void_shipping`
	Disposition	`in_transit`	`in+progress`
	Source	`owning_party`: GLN of Company X	`owning_party`: GLN of Company X
	Destination	`owning_party`: GLN of Company Y	`owning_party`: GLN of Company Y

Note that the event time, read point, business location, and disposition reflect the process of voiding the shipment: the event time is the date/time the shipment was voided, the business location is the warehouse reflecting the location of serial number 103 after shipmen is voided, and the disposition is "in progress" as it would if serial number 103 had not been shipped. However, the source and destination is the same in the `void_shipping` event as in the original shipping event, reflecting the context for the voided business transfer.

5.9.3 Example 3: Declaring a prior event to be in error, with no corrective event

In this example, Company X records an EPCIS event to assert that serial number 101 of product X was destroyed. This event is an Object Event with action = DELETE. Later it is discovered that serial 101 is still in storage, not destroyed. An ordinary event cannot be used to amend the history, because the semantics of action DELETE for an Object Event specify that "the objects … should not appear in subsequent events."

The remediation is to issue an error declaration event. This looks just like the original, erroneous event, but with the addition of an error declaration section.

Both events together look like this:

Table 5-21 Example of correcting an error by adding an error declaration event.

Dim	Data Element	V1	V2
	Description	Destroy one instance of Product X, not realising that this instance was not destroyed	Additional event to assert that the first event is in error
	Event Type	Object Event	Object Event
	Action	`DELETE`	`DELETE`
Error Declaration	**Declaration Time**		17 July, 2pm
	Reason		`did_not_occur`
When	**Event Time**	15 July, 10am	15 July, 10am
What	**EPC List**	GTIN X, Serial 101	GTIN X, Serial 101
Where	**Read Point**	SGLN of warehouse	SGLN of warehouse
	Business Location	(omitted)	(omitted)
Why	**Business Step**	`destroying`	`destroying`
	Disposition	`destroyed`	`destroyed`

5.9.4 Example 4: Declaring a prior event to be in error, with a corrective event

Company X records an EPCIS event asserting that several products have been shipped, indicating Purchase Order 123 as a business transaction in the "why" dimension. Company Y receives the products and records a receiving event. Only then it is discovered that the purchase order reference in the shipping event is wrong: it says PO 456 instead of 123. This could be remediated using ordinary EPCIS events by Company X recording a "void shipment" event followed by a "shipping" event with the correct PO #. But this is rather undesirable from the perspective of the overall trace, especially given that there is already a receiving event

The remediation is to issue an error declaration event together with a corrective event. The error declaration looks just like the original, erroneous event, but with the addition of an error declaration section. The corrective event is a corrected version of the original event. Optionally, the corrective event can be given a unique event ID, and referenced from the error declaration event.

All three events together look like this:

Table 5-22 Example of correcting an error by adding an error declaration event.

Dim	Data Element	V1	V2	V3
	Description	Ship products, not realising that the PO number in the business transaction section is incorrect	Additional event to assert that the first event is in error	Corrected shipping event
	Event Type	Object Event	Object Event	Object Event
	Action	`OBSERVE`	`OBSERVE`	`OBSERVE`
	Event ID			UUID 692…6bd
Error Declaration	**Declaration Time**		17 July, 1pm	
	Reason		`incorrect_data`	

Dim	Data Element	V1	V2	V3
	Corrective Event IDs		UUID 692…6bd	
When	**Event Time**	15 July, 10am	15 July, 10am	15 July, 10am
What	**EPC List**	GTIN X, Serial 101, 102, 103	GTIN X, Serial 101, 102, 103	GTIN X, Serial 101, 102, 103
Where	**Read Point**	SGLN of warehouse	SGLN of warehouse	SGLN of warehouse
	Business Location	(omitted)	(omitted)	(omitted)
Why	**Business Step**	`shipping`	`shipping`	`shipping`
	Disposition	`in_transit`	`in_transit`	`in_transit`
	Business Transactions	PO #456	PO #456	PO #123

5.9.5 Timing of capturing error declaration and corrective events

As the example in section 5.9.4 illustrates, an error declaration is sometimes accompanied by one or more corrective events. It is important that an EPCIS Accessing Application that receives event data be aware of the error declaration if it sees the corrective event(s), because otherwise the application may see an inconsistency between the original (erroneous) event and the corrective events.

For this reason, it is important that corrective event(s) are not sent to an EPCIS Capture Interface prior to sending the error declaration event. On the other hand, if the error declaration event makes a forward reference to the corrective event(s) using the `correctiveEventIDs` field, then the corrective events must be known to the EPCIS Capturing Application at the time the error declaration event is generated. The recommended way to address both of these concerns at once is for the error declaration and associated corrective event(s) to be captured *at the same time*; that is, within the same event list in the document delivered to the EPCIS Capture Interface.

Note that the above considerations are not related to the event time or declaration time fields of the events concerned. The declaration time of the error declaration is the date and time at which the error declaration is made, the event time of the error declaration is identical to the event time of the original erroneous event (usually preceding the declaration time, unless the event time was one of the things that was wrong with the original event), and the event time of the corrective event(s) is the date and time at which the event actually occurred (usually the same as the event time of the original event, unless the event time was one of the things that was wrong with the original event).

5.9.6 Querying for events in the presence of errors and corrections

An error declaration event is constructed by including an `ErrorDeclaration` section. Specifically, given Event E1, an error declaration event E2 whose effect is to declare the assertions of E1 to be in error is an event structure whose content is identical to E1, but with the `ErrorDeclaration` element included. For example, the error declaration for the "destroying" event in Example 3 is also an Object Event with action = DELETE, but with the `ErrorDeclaration` element included. In general, to declare event E to be in error, a new event is recorded that is identical to event E except that the `ErrorDeclaration` element is also included (and the record time will be different).

There are three reasons why error declaration events in EPCIS are expressed this way. One, an event ID is not required to indicate the erroneous event, which in turn implies it is not necessary to include an event ID on every event to provide for possible error declaration in the future. Event IDs are available to link an error declaration event to a corrective event, but it is never necessary to use event IDs. Two, any EPCIS query that matches an event will also match an error declaration for that event, if it exists. This means that EPCIS Accessing Applications require no special logic to become aware of error declarations, if they exist. Three, if an EPCIS Accessing Application receives an error declaration event and for some reason does *not* have a copy of the original (erroneous) event, it is not necessary to retrieve the original event as every bit of information in that event is also present in the error declaration event.

5.10 Association Events

The EPCIS Event Type `AssociationEvent` as introduced as of EPCIS 2.0 enables organisations to capture associations of physical objects that are more permanent compared to temporary relationships that are captured through e.g. packing or loading events. This is useful to have precise visibility on which items were built into which products, assemblies, or assets.

For instance, applying Association Events is applicable in the following situations:

- installation of a sensor device into a reusable plastic tray
- construction of a rail wagon (which is made up by axles, bogies, roof components, brake systems, buffers, etc.)
- removing a (defect) component from an assembly

Prior to EPCIS 2.0, companies had to use AggregationEvents for such use cases. However, the EPCIS standard allows to capture an AggregationEvent with an empty childEPCs and/or childQuantityList element when the action value is DELETE. Note though that plastic trays or rail waggons can also be used for more temporary aggregations such as in the course of loading or packing events. If an EPCIS-based visibility system used AggregationEvents also for the construction of transport units and captured an unloading or unpacking event with an empty childEPCs and/or childQuantityList element, it would mean that not just the packed or loaded objects were disaggregated from these transport units, but all the items the transport units themselves are made of, too.

5.10.1 Example 1: Installing components/assemblies into larger items

For illustration purposes, presume a pool operator of reuseable plastic trays wants to properly document the installation of sensor devices (with e.g. a built-in GPS module and temperature sensor) into its trays. The reason could consist in the need to effectively identify all assets that are equipped with specific sensor devices/models in case the latter were not exactly calibrated. In addition, information on built-in sensor devices may also be enquired by customs authorities. In such a scenario, the installing EPCIS event could be modelled as follows:

EPCIS dimension	Data Element	V1
	Description	Installing a sensor device in a reusable plastic tray
	Event Type	Association Event
	Action	ADD
When	`eventTime`	12 October, 08:45 am
What	`parentID`	GRAI of tray
	`childEPCs`	GIAI of sensor device
	`readPoint`	GLN of maintenance area
	`bizStep`	`installing`

5.10.2 Example 2: Installing components/assemblies into physical locations

The `AssociationEvent` is the only EPCIS event type where it is permissible to populate the `parentID` field with a physical location identifier. This feature is especially relevant for companies that need to document which particular item became an integral part of a physical location.

In a way, it is similar to the previous example, but in this situation, items are integrated into buildings or rooms rather than larger assemblies. Note that an AssociationEvent is not applicable if, for instance, a room is equipped with pieces of furniture - in such a case, the association is not permanent and organisations should use an `ObjectEvent` instead.

Taking the example of a company that equips a cold storage room with one or several temperature sensor devices, the corresponding EPCIS event may be modelled as follows:

EPCIS dimension	Data Element	V1
	Description	Installing a sensor device in a reusable plastic tray
	Event Type	Association Event
	Action	ADD
When	`eventTime`	14 October, 10:55 am
What	`parentID`	GLN of cold storage room
	`childEPCs`	GIAI(s) of sensor device(s)
	`readPoint`	GLN of maintenance area
	`bizStep`	`installing`

5.11 Sensor-based quality data

To improve e.g. patient safety, consumer protection, supply chain visibility and food safety, there is a growing need to capture and share sensor data. The Sensor Element, introduced as of EPCIS 2.0, allows organisations to provide trading partners such data in a standardised manner – for instance, if they want to prove that goods never exceeded a specific sensor property value during the time they had custody of these items.

It is of paramount importance that EPCIS is not meant to transmit raw sensor data dumps. Rather, its added value consists in the ability to provide applications business-oriented, aggregated sensor data. For example, retailers typically are just interested in knowing whether they can put received goods on their shelves or not – in other words, if products were handled within an agreed temperature range. They are not concerned about discrete temperature values at specific timestamps. Therefore, even though the EPCIS data model would *theoretically* allow to accommodate time-stamped sensor data, organisations should model EPCIS events transmitting sensor data very carefully. (Note: even if there is a need to access the original sensor data underlying a given EPCIS event, organisations can use the standard field rawData to point to that data without having to blow up the EPCIS event itself.)

5.11.1 Sensor example 1: Control/prove temperature compliance

Suppose an organisation that trades temperature-sensitive goods (e.g. cheese, wine, pharmaceutical products) has set up the necessary hardware to capture both the identities as well as the temperature values of items when the latter are in the company's custody.

Now, if this organisation wants to provide that data to internal or external stakeholders (e.g. the company's quality assurance department or trading partners that wish to ascertain if specific items were handled/transported properly), it makes a lot of sense to use a standard format from the outset.

Typical critical tracing events accommodating sensor data can easily be modelled as EPCIS events. Following the usual approach, a visibility data matrix could look like this (the table focusses on the relevant excerpt of the overall chain of events):

EPCIS dimension	Data Element	V1	V2	V3	V4
	Description	Move logistics unit to interim storage room	Move logistics into cold storage room	Move logistics out of cold storage room	Daily sensor reporting of cold storage room
	Event Type	Object Event	Object Event	Object Event	Object Event
	Action	`OBSERVE`	`OBSERVE`	`OBSERVE`	`OBSERVE`
When	`eventTime`	15 June, 08:00 am	15 June, 08:15 am	15 June, 05:45 pm	15 June, 11:59 pm
What	`epcList`	SSCC of logistics unit	SSCC of logistics unit	SSCC of logistics unit	
	`readPoint`	GLN of receiving area	GLN of interim storage room	GLN of cold storage room	GLN of cold storage room
	`bizLocation`	GLN of interim storage room	GLN of cold storage room	GLN of shipping area	
	`bizStep`	`storing`	`storing`	`storing`	`Sensor_reporting`
	`sensorElement`				
	`sensorReport`				
	`startTime`	15 June 07:55 am	15 June 08:10 am	15 June 05:35 pm	14 June 11:59 pm
	`endTime`	15 June 07:59 am	15 June 08:14 am	15 June 05:55 pm	15 June 11:59 pm
	`type`	`Temperature`	`Temperature`	`Temperature`	`Temperature`
	`minValue`	12	12.1	9.2	9.1
	`maxValue`	12.1	12.2	9.2	9.4
	`uom`	`CEL`	`CEL`	`CEL`	`CEL`

On this basis, the organisation has an unbroken chain of events documenting the condition of an individual item, beginning from when it was relocated from the receiving area to an interim storage room (V1), when it was moved in and out of the cold storage room (V2 and V3), and while it was residing in the cold storage room (V4).

As to V4, note that as of EPCIS/CBV 2.0, a CBV-compliant EPCIS event is allowed to have an empty WHAT dimension, if a non-empty Sensor Element is present. In such a case, the object of observation is the physical location indicated in the WHERE dimension (i.e. populating either readPoint or bizLocation). Also, V4 leverages bizStep 'sensor_reporting' which is an appropriate choice when no actual business process step is ongoing.

With regard to designing the HOW dimension, the organisation has ample flexibility. For instance, they *could* have included a pointer to the underlying raw sensor data (rawData), indicated the ID of the respective sensor devices (deviceID) or inserted a reference to the meta data of a given sensor device (deviceMetadata). For simplicity, we assume that the business need consists in controlling that the ambient temperature did not exceed a specific minimum or maximum value. For this purpose, the company can get by with a very concise set of attributes: the start and end time of a related sensor reading as well as the highest and lowermost temperature value within that period, expressed in degree Celsius.

In this context, the company could also have chosen another appropriate unit of measure listed in UN/ECE Recommendation 20 (i.e. Kelvin, degrees Fahrenheit or Rankine).

For convenience and to ease implementation, GS1 provides an Open Source library to automatically convert between any quantitative value of a given property type (e.g. temperature). The library is available at **https://ref.gs1.org/tools/UnitConverterUNECERec20/** .

5.11.2 Sensor example 2: Exception notification

Presume a company wishes to trigger processes (adjust settings of an environmental control system, alert an employee, etc.) when a certain condition (e.g. a temperature excursion) occurs.

Pursuing the example from the previous section, a company may want to trigger an alert message to the warehouse manager in case the temperature in the cold storage room falls below or exceeds a predefined threshold (e.g. < 8 ° CEL and > 15 ° CEL). The company also wants to store that information in their Quality Management System as well as provide that to an external solution provider which is in charge of maintaining the cold storage room's technical infrastructure.

In such a setting, the 'alert' EPCIS event could be modelled as follows:

Event dimension	Data Element	V1
	Description	Exception notification for temperature excursion
	Event Type	Object Event
	Action	OBSERVE
When	eventTime	23 June, 11:19 am
Where	readPoint	GLN of cold storage room
Why	bizStep	sensor_reporting
How	sensorElement	
	sensorMetadata	
	bizRules	GDTI GS1 DL URI
	sensorReport	
	type	Temperature
	value	15.1
	uom	CEL
	sensorReport	
	exception	ALARM_CONDITION
	uriValue	URI, e.g. https://example.com/alarmCodes/temperatureExceeded

In contrast to the previous example, the event accommodates the (optional) sensorMetadata field, which in turn contains a reference (the Web URI is a valid GS1 Digital Link URI leveraging a custom (here: "example.com") domain, 253 denotes the GS1 Application Identifier for the Global Document Type Identifier) to an electronic document including the business rule(s) upon which the EPCIS event was captured. The company may decide to also insert additional attributes such as deviceID or deviceMetadata into this element, if applicable.

Apart from the actual temperature value (exceeding the predefined threshold), the sensorElement contains a second sensorReport element accommodating an alarm value, expressed as a URI. The latter consists of a custom value - a future GS1 working group may define standard vocabulary for alarm/error code values for this application domain.

5.11.3 Sensor example 3: Condition monitoring and tracking of intermodal transports

Goods are often transported through several modes of transport, e.g. in sea containers, trucks or railway carriages. To allow a company to verify whether their products are properly transported, and to maintain an overview of the areas a container vessel traversed, it is advisable for the respective logistics/transport service providers to supply the corresponding visibility event data in a standardised manner.

For instance, if an organisation is interested to ascertain that their products were not exposed to a certain level of air humidity during transport as well as the approximate sea transport route, the following EPCIS event sequence would make sense:

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	Description	Pack products into logistics unit	Load logistics unit onto sea container	Load sea containers onto truck	Truck arrival at port	Unload sea containers from truck	Load sea containers onto vessel	Vessel departure from port	Daily sensor reporting of sea container	Daily vessel report with 4-hourly geo positions	Daily sensor reporting of sea container
	Event Type	Aggregation Event	Aggregation Event	Aggregation Event	Object Event	Aggregation Event	Aggregation Event	Object Event	Object Event	Object Event	Object Event
	Action	ADD	ADD	ADD	OBSERVE	DELETE	ADD	OBSERVE	OBSERVE	OBSERVE	OBSERVE
When	eventTime	24 June, 08:00 am	24 June, 09:15 am	24 June, 09:45 am	24 June, 02:20 pm	24 June, 02:55 pm	24 June, 05:11 pm	25 June, 04:00 am	24 June, 11:59 pm	25 June, 11:59 pm	25 June, 11:59 pm
What	epcList				GIAI of the truck			IMO Vessel Number of ship	BIC of sea container	IMO Vessel Number of ship	BIC of sea container
	parentID	SSCC of logistics unit	BIC of sea container	GIAI of the truck		GIAI of the truck	IMO Vessel Number of ship				
	childEPCs	SGTINs of products	SSCC of logistics unit	BIC of sea container		BIC of sea container	BIC of sea container				

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
Where	readPoint	GLN of warehouse	GLN of warehouse	GLN of warehouse	GLN of port	GLN of port	GLN of port	GLN of port			
Why	bizStep	packing	loading	loading	arriving	unloading	loading	departing	sensor_reporting	sensor_reporting	sensor_reporting
How	sensorElement										
	sensorMetadata										
	startTime							23 June 11:59 pm		24 June 11:59 pm	
	endTime							24 June 11:59 pm		25 June 11:59 pm	
	sensorReport										
	type							Temperature		Temperature	
	minValue							8.1		5.6	
	maxValue							21.8		14.9	
	uom							CEL		CEL	
	sensorReport										
	type							AbsoluteHumidity		AbsoluteHumidity	
	minValue							6.1		4.6	
	maxValue							8.2		3.3	

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	uom							A93		A93	
	sensorElement										
	sensorMetadata										
	time								25 June 02:00 am		
	rawData								URI, e.g. https://example.org/8004/401234599999		
	sensorReport										
	type								`Latitude`		
	stringValue								53.553747		
	sensorReport										
	type								`Longitude`		
	stringValue								8.562372		
	sensorElement										
	sensorMetadata										
	time								25 June 06:00 am		
	sensorReport										

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	type								Latitude		
	stringValue								53.882318		
	sensorReport										
	type								Longitude		
	stringValue								8.099310		
	sensorElement										
	sensorMetadata										
	time								25 June 10:00 am		
	sensorReport										
	type								Latitude		
	stringValue								54.172892		
	sensorReport										
	type								Longitude		
	stringValue								7.094428		
	sensorElement										
	sensorMetadata										
	time								25 June 02:00 pm		

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	sensorReport										
	type								Latitude		
	stringValue								54.389794		
	sensorReport										
	type								Longitude		
	stringValue								5.753072		
	sensorElement										
	sensorMetadata										
	time								25 June 06:00 pm		
	sensorReport										
	type								Latitude		
	stringValue								54.790116		
	sensorReport										
	type								Longitude		
	stringValue								3.407863		
	sensorElement										
	sensorMetadata										

Event dimension	Data Element	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	time								25 June 10:00 pm		
	sensorReport										
	type								`Latitude`		
	stringValue								56.196056		
	sensorReport										
	type								`Longitude`		
	stringValue								1.490934		

Note that though further appropriate EPCIS events (e.g., shipping, receiving) were omitted for simplicity reasons, the above sequence of events enables the organisation to obtain a complete view of how an individual item was transported.

The Aggregation Events (V1, V2, V3, V5 and V6) allow for precise knowledge which individual products were, at which point in time, packed into which containers and hauled with which means of transport (here: a truck and a vessel).

To determine whether temperature and air humidity (uom "A93" stands for gram per cubic metre, a possible unit for measuring absolute humidity) are below an acceptable level, the accessing application only needs to query for the corresponding daily sensor reporting events via the data owner's EPCIS repository (V8 and V10).

Event V9 illustrates the use of sensor-related standard extension fields to transmit geographic positions of a given item. In this case, the EPCIS capturing application triggers an event at the end of each day, thereby inserting the latitude/longitude values in 4-hour intervals. If an accessing client is interested in more granular data, the event also includes a Web URI (which again is a valid GS1 Digital Link Web URI - thereby, '8004' is the GS1 AI for a Global Individual Asset Identifier) pointing to the underlying raw sensor data.

5.11.4 Sensor example 4: Condition monitoring and tracking of intermodal transports

For consumer safety reasons or due to legal requirements, many organisations need to conduct quality controls. For instance, common practice is to take a control/random sample at goods receipt. As of EPCIS 2.0, organisations can properly capture and document the concentration of potentially harmful bacteria and other microorganisms. What is more, they can also capture the concentration of any chemical substance.

For illustration purposes, let us assume that a retailer wants to document the concentration of Shigella (bacteria that include known pathogens) as well as sugar in a batch/lot of apples. Further, the retailer wants to capture the ID of the device with which the quality control is accomplished (so that in case the latter turns out not to be properly calibrated, the retailer is able to react accordingly). With that in mind, an EPCIS inspecting event could be designed as follows:

Event dimension	Data Element	V1
	Description	Fresh fruits quality inspection
	Event Type	Object Event
	Action	OBSERVE
When	eventTime	10 August, 08:10 am
What	quantityList	LGTIN of batch/lot of food
	readPoint	GLN of cold storage room
	bizStep	inspecting
	disposition	conformant
	bizTransactionList	
	_bizTransactionID type:testprd	GDTI of test procedure
	_bizTransactionID type:testres	GDTI or test result
	sensorElement	
	sensorMetadata	
	deviceID	GIAI (EPC URI or GS1 DL URI)
	sensorReport	
	type	Dimensionless
	microorganism	https://wwww.ncbi.nlm.nih.gov/1118236 TBC
	value	18
	uom	CFU/ml *Pending UN/CEFACT update to* [CEFACT20]
	sensorReport	
	type	Dimensionless_concentration

Event dimension	Data Element	V1
	chemicalSubstance	https://identifiers.org/inchikey:CZMRCDWAGMRECN-UGDNZRGBSA-N
	value	10.1
	uom	J18

The above example can be easily applied to all other use cases in which there is a need to capture the concentration of either chemical substances or microorganisms in the objects indicated in the What dimension. Note that for populating the first one, the CBV specifies to use the International Chemical Identifier Key URI. For the second one, it defines to use the NCBI (National Center for Biotechnology Information) Web URI. Both URI schemes ensure uniqueness and are actually resolvable, i.e. can return further information on the respective organic or inorganic subjects.

The uom "J18" is the UN/CEFACT common code of degree Brix, a unit of proportion used in measuring the dissolved sugar-to-water mass ratio.

5.12 End-of-Life Events for Instance-Level Identification

An end-of-life event for an active instance-level identification becomes necessary when the instance-level identification must be inactivated and disassociated from the physical object, in order to remove the specific instance of the physical object from circulation (e.g., because the object has been physically destroyed or consumed). An end-of-life event is an Object Event with Action DELETE. Since bizLocation is the location where the object is presumed to be following the event, the bizLocation for an object which has undergone an end-of-life event is undefined and shall be omitted from the event.

Several business steps from the CBV signify end-of-life events for the instance-level identification of an object: decommissioning, destroying, and dispensing.

When an object's instance-level identifier is decommissioned:

- the instance-level identifier ceases to exist even though the object may still physically exist.
- the identifier's link to the physical object ceases to exist.

A decommissioned identifier should not appear in EPCIS events thereafter, nor can a decommissioned identifier be reactivated.

~~An object whose identifier has been decommissioned can be newly commissioned with a new instance-level identifier (e.g., using an EPCIS Transformation Event).~~

5.13 Inferred completeness

EPCIS events are typically used to record structured event data about activities in the real world, including observations and completion of business steps, often triggered by the scan of one or more barcodes or the detection of one or more RFID tags.

There are occasions when an EPCIS event expresses the identifiers of **objects whose barcodes or RFID tags have not actually been observed at that point in time but are considered to be present**, based on prior event data or other information, combined with physical inspection that tamper-evident seals have not been broken. Such **practice of inference** occurs in supply chains and is recognised and accepted by some legislation concerned with traceability.

However, discrepancies between physical and electronic capture of packing events can occur, so in order to resolve such discrepancies, it can be very helpful if application software is able to distinguish between such events based on inference, versus those events where each mentioned identifier of a physical object has been positively verified through a scan of a barcode or RFID tag or other AIDC data carrier.

Although completeness_inferred and completness_verified are defined within CBV 2.0, completeness_inferred cannot be used as the value of persistentDisposition within an AggregationEvent, since persistentDisposition is not defined for AggregationEvent -

only for ObjectEvent and the output of TransformationEvent. Expressing a persistentDisposition of completeness_inferred for an aggregation would itself be problematic because of the need to remember to explicitly unset this for the parent as soon as one child of the aggregation has been disaggregated.

A simpler approach is not use completeness_inferred or completeness_verified within disposition or persistentDisposition and instead use a dedicated field within a custom namespace to indicate an event where some IDs of physical objects are based on inference, e.g. setting a custom field such as gs1ushc:completenessInferred = true (with a default value of gs1ushc:completenessInferred = false, if the custom field is absent, indicating that all IDs of physical objects mentioned within the event were positively verified).

This approach using a Boolean field is simpler to understand and implement and does not require anything to be subsequently unset.

6 Sharing EPCIS Data

EPCIS data records the *what*, *when*, *where*, *why* and (where applicable) *how* of business processes in which physical or digital objects are handled. Such data may be used by many different business applications. This section discusses some of the practical aspects of sharing EPCIS data, both sharing with applications within one organisation's four walls, and sharing between trading partners to achieve overall supply chain or ecosystem process visibility.

6.1 EPCIS Queries

EPCIS Accessing Applications obtain EPCIS events from an EPCIS Repository by means of an EPCIS Query. An EPCIS Query is a set of event-matching criteria specified by the application; the EPCIS Repository responds to a query by retrieving all EPCIS events that match the specified criteria.

The EPCIS Standard, section 8.2.7.1, defines over 40 different criteria that can be used to construct a query for event data. These criteria can be used alone or in combination. Each criterion has a "parameter name" specified in the EPCIS standard, and most take a "parameter value" that further defines how the criterion is to be applied. The following table lists some of the commonly used criteria; see the EPCIS Standard, Section 8.2.7.1, for the complete list:

Table 6-1 Selected EPCIS Query Criteria

Query Criterion Parameter Name	Query Criterion Parameter Value	Description
eventType	One or more event types: ObjectEvent, AggregationEvent, TransactionEvent, TransformationEvent or AssociationEvent	Matches events whose event type is one of the event types named in the parameter value.
EQ_action	One or more action values: ADD, OBSERVE, or DELETE	Matches events that include an action and where the action is one of the actions named in the parameter value
GE_eventTime	A date/time value (including a time zone specifier)	Matches events whose event time is on or after the date/time specified in the parameter value.
LT_eventTime	A date/time value (including a time zone specifier)	Matches events whose event time is prior to the date/time specified in the parameter value.
MATCH_anyEPC	One or more instance-level identifiers, or patterns matching instance-level identifiers	Matches events whose *what* dimension contains at least one instance-level identifier that matches one of the identifiers or patterns specified in the parameter value. MATCH_anyEPC looks for matching instance-level identifiers anywhere in the *what* dimension. Other query criteria are defined in the EPCIS standard that match specific parts of the *what* dimension; e.g. matching just the parent of an aggregation event but not children.

Query Criterion Parameter Name	Query Criterion Parameter Value	Description
`EQ_readPoint`	One or more location identifiers	Matches events whose read point (in the *where* dimension) is equal to one of the location identifiers specified in the parameter value.
`EQ_bizLocation`	One or more location identifiers	Matches events whose business location (in the *where* dimension) is equal to one of the location identifiers specified in the parameter value.
`EQ_bizStep`	One or more business step identifiers	Matches events whose business step (in the *why* dimension) is equal to one of the business step identifiers specified in the parameter value.
`EQ_disposition`	One or more disposition identifiers	Matches events whose disposition (in the *why* dimension) is equal to one of the disposition identifiers specified in the parameter value.
`EQ_bizTransaction_`*`XXX`*	One or more business transaction identifiers	Matches events that contain a business transaction (in the *why* dimension) whose business transaction type is *XXX* and whose business transaction identifier is equal to one of the identifiers specified in the parameter value. To use this parameter, the business transaction type replaces *XXX* in the parameter name; see below.
`EQ_`*`XXX`*	One or more strings	Matches events having an extension element named *XXX*, where the contents of that extension element is a string matching one of the strings specified in the parameter value. To use this parameter, the XML element name of the extension replaces *XXX* in the parameter name; see below.

A single query may include more than one criterion, in which case events must match *all* criteria to be included in the result. For example, a query that includes both the `GE_eventTime` and the `MATCH_epc` criteria will match only those events that occur on or after the specified event time *and* which contain one of the specified instance-level identifiers.

To answer a business question, first the information need must be identified, then analysed to determine what EPCIS events contain the needed information. Then, a suitable EPCIS query can be formulated. The following table illustrates how that would be done for several typical examples.

Table 6-2 Examples of Business Information Needs and Corresponding EPCIS Query Criteria

Business Information Need	Relevant EPCIS Events	EPCIS Query Criteria
Confirm that EPC XXX is valid, and determine the date it was created and associated properties such as the lot, expiration date, etc.	The EPCIS event for the commissioning step bearing EPC XXX. The EPC is valid if this event exists. This event also includes the event time and instance/lot master data that answers the other questions.	`MATCH_epc`: XXX `EQ_bizStep: urn:epcglobal:cbv:bizstep:commissioning`
Find out all of the products that were received at loading dock door #23 on March 15, 2014	All EPCIS events with business step receiving, whose read point is dock door #23, with event times on the desired date	`GE_eventTime: 2014-03-15T00:00:00Z` (midnight UTC on 15 March 2014) `LT_eventTime: 2014-03-16T00:00:00Z` (midnight UTC on 16 March 2014) `EQ_readPoint:` (SGLN identifier for dock door #23) `EQ_bizStep: urn:epcglobal:cbv:bizstep:receiving`

Business Information Need	Relevant EPCIS Events	EPCIS Query Criteria
Find out the specific serial numbers that were shipped to fulfill purchase order #559	The EPCIS event for the shipping step having a transaction identifier for PO #559. The CBV business transaction type for PO is `urn:epcglobal:cbv:btt:po`. The PO number is encoded using the CBV template for creating a business transaction identifier using a GLN; in this example assume the GLN is 0123456789012	`EQ_bizTransaction_urn:epcglobal:cbv:btt:po: urn:epcglobal:cbv:bt:0123456789012:559`

Identify all logistics units of a given organisation that were transported at a temperature of 15.5 ° CEL or more	All EPCIS events with business step `transporting` that have a `sensorReport` element of type `Temperature`, which are populated with SSCCs featuring a specific GCP, with a corresponding quantitative value equal to or greater than 15.5 CEL	`EQ_bizStep: transporting`
		`MATCH_epc`: SSCC ID Pattern
		`EQ_type: Temperature`
		GE_value_CEL: 15.5

6.2 Query Modes: Pull vs Push

An EPCIS query is used to transfer EPCIS events from an EPCIS repository to an application or trading partner that needs those events. There are different ways in which the transfer can be triggered.

- **Pull**: This method of transfer involves a request/response pattern. The application or trading partner issues a request to an EPCIS repository containing EPCIS query criteria, and the EPCIS repository responds with the EPCIS events that match the criteria.
- **Push**: This method of transfer involves a one-way message: the EPCIS repository simply delivers one or more EPCIS events to an application or trading partner that needs them. There are two variations to this theme:
  - ***Pre-arrangement***: The sending and receiving party have agreed in advance, by some means outside of the scope of the EPCIS standard, what data the receiving party needs and under what conditions. The sending party delivers events when it sees fit based on that prior arrangement.
  - **Subscription**: The receiving party issues a *standing query* to the sending party to express an ongoing information need. A standing query includes EPCIS query criteria (just as in the "pull" method) along with description of the conditions that will trigger the delivery of events. These conditions could be a regular schedule (e.g., daily at 3am) or some other triggering event. Each time the triggering condition occurs, the sending party evaluates the query criteria and delivers any new events that match the criteria (new compared to the last time the subscription was triggered).

In both "push" variations, EPCIS events are delivered in a one-way communication from sender to receiver; the difference is that in the pre-arrangement variation the sender is in full control of what data is sent whereas in the subscription variation the receiver gets to express its needs via the standing query. In all method, "push" and "pull", the sender ultimately has control over what data is sent, as described in section 6.7.

In designing an overall business process that involves the flow of EPCIS data, different query modes may be used to meet differing requirements. Generally speaking, "push" methods are often used when there is a recurring predictable need for transfer of EPCIS data, and "pull" methods are used when transfer is needed unpredictably or only on an exception basis. The following table shows examples under which each variation might be appropriate:

Table 6-3 Example Business Scenarios and Corresponding Likely EPCIS Query Modes

Example Business Scenario	Query Mode	How the Query Mode is Employed
GTIN X, Lot Y has been recalled: Manufacturer XYZ needs to find out if Retailer ABC has received any of that product.	Pull	Manufacturer XYZ issues a request to ABC's EPCIS repository, querying for all events containing GTIN X, Lot Y in the *what* dimension and "receiving" in the business step.
In compliance with local regulation, Distributor PQR needs to send information about each serial number pharmaceutical product it ships to Pharmacy ABC, within one hour of shipment.	Push via pre-arrangement	PQR and ABC have agreed to this in advance and ABC has provided PQR with the address where such messages are to be directed. Each time PQR makes a shipment of pharmaceuticals to ABC, its EPCIS Repository sends a message to ABC containing the EPCIS events having the serialised GTINs of the pharmaceuticals in the *what* dimension and "shipping" in the business step.
In order to prepare for the special handling required, Retailer ABC wants to be notified whenever Manufacturer XYZ sends it a shipment containing Product X, which contains hazardous materials	Push via subscription	Retailer ABC issues a standing query whose criteria match all EPCIS events containing GTIN X in the *what* dimension, "shipping" in the business step, and its GLN in the destination list, to be triggered on a daily basis.

6.3 The EPCIS Query Control Interface

The EPCIS standard provides a standardised interface through which an EPCIS accessing application or trading partner may interact with an EPCIS repository. Through this interface, an application or trading partner may issue a "pull" query, set up a "push" subscription, and more.

The following table summarises the operations available through the interface:

Table 6-4 EPCIS Query Control Interface Operations

Operation	Description	Request (from EPCIS accessing application or trading partner)	Response (by EPCIS Repository)
`poll`	Execute a "pull" query	EPCIS query criteria	EPCIS events matching the query criteria
`subscribe`	Set up a "push" subscription	A subscription ID chosen by the requestor, EPCIS query criteria, triggering conditions, and an address for delivery of standing query results	An acknowledgement. Subsequently, the EPCIS repository will deliver to the specified address events matching the criteria when the trigger conditions occur
`unsubscribe`	Cancel a previous subscription	The subscription ID previously used to establish the subscription	An acknowledgement
`getSubscriptionIDs`	Find out what subscriptions are active	[no contents]	A list of subscription IDs previously subscribed by the requestor
`getStandardVersion`	Find out what version of the EPCIS standard is supported by the EPCIS repository	[no contents]	1.0 or 1.1, depending on what version the repository supports
`getVendorVersion`	Find out vendor-specific information about the EPCIS repository implementation	[no contents]	A string defined by the EPCIS Repository vendor.

Operation	Description	Request (from EPCIS accessing application or trading partner)	Response (by EPCIS Repository)
`getQueryNames`	Find out what types of EPCIS queries are supported by the EPCIS repository	[no contents]	A list of queries supported by this EPCIS Repository. This always includes SimpleEventQuery as defined by the EPCIS standard and may include SimpleMasterDataQuery. It may also include other available queries that are vendor-specific.

The EPCIS standard defines XML representations for each request and response message that is used in the EPCIS query interface.

6.4 Choreography Models: Sharing Data across a Supply Chain

When two trading partners share EPCIS information with each other, the flow of information is straightforward: each partner has an established business relationship with the other, and they agree on what data to share and what query mode to use.

The situation is more complex in an ecosystem having many trading partners. Each party may be trading with many others, and each such trading relationship may require the exchange of EPCIS data. Moreover, it may be necessary for one party to share EPCIS data with another party with whom there is not a direct trading relationship; for example, if A sells to B and B sells to C, there may be a need for A and C to share EPCIS data to get a complete picture of the supply chain, even though A and C do not have a direct trading relationship.

A core principle for managing this complexity is to *separate content from choreography*. What this means is that the *content* of EPCIS data – the specific business steps that require visibility, the EPCIS events that will used to record the completion of those steps, and the detailed contents of those events – should be designed according to the methods described earlier in this guideline (sections 3, 4, and 5). These methods focus on accurately modelling the *what*, *when*, *where*, *why* and, where applicable, *how* information for each business step. Separately from that, trading partners can decide when and how the data will move from one trading partner to another – this is called the *choreography*. Choreography decisions include: where will data reside, what will trigger the communication of data from one party to another, will push or pull modes be used, what networking technology will be used, and so forth. By separating content from choreography, the choreography can adapt to changes in the size of the trading ecosystem and evolution of technology, while the design of the EPCIS content stays the same.

There are many possible approaches to choreography. Many of these approaches fall into one of the following three broad categories:

- **Centralised Choreography**: In these models, EPCIS events from multiple parties in the supply chain are sent to a shared repository. To get an overall view of the supply chain, it is only necessary to query the shared repository.
- **Distributed Query Choreography**: In these models, each party that captures EPCIS data keeps that data in its own repository. When another party needs an overall view of the supply chain, it must locate and query all of the other parties who may have relevant data within their respective repositories.
- **Distributed Push Choreography**: In these models, each party that captures EPCIS data keeps that data in its own repository. But rather than waiting for another party to query for that data, the capturing party sends (pushes) its data to other parties in the supply chain who are likely to need that data. Often the push of data follows the same path as the physical or digital objects; e.g. if a Party A ships goods to Party B, it also sends its EPCIS data to Party B.

The following sections illustrate examples of these three approaches in more detail. Throughout these sections, a scenario is illustrated in which Party A ships goods to Party B who ships goods to Party C, and upon receipt Party C would like to examine the upstream EPCIS events from A and B.

Differences between choreography approaches are illustrated through the following four questions:

- Questions for the *producers* of EPCIS event data:

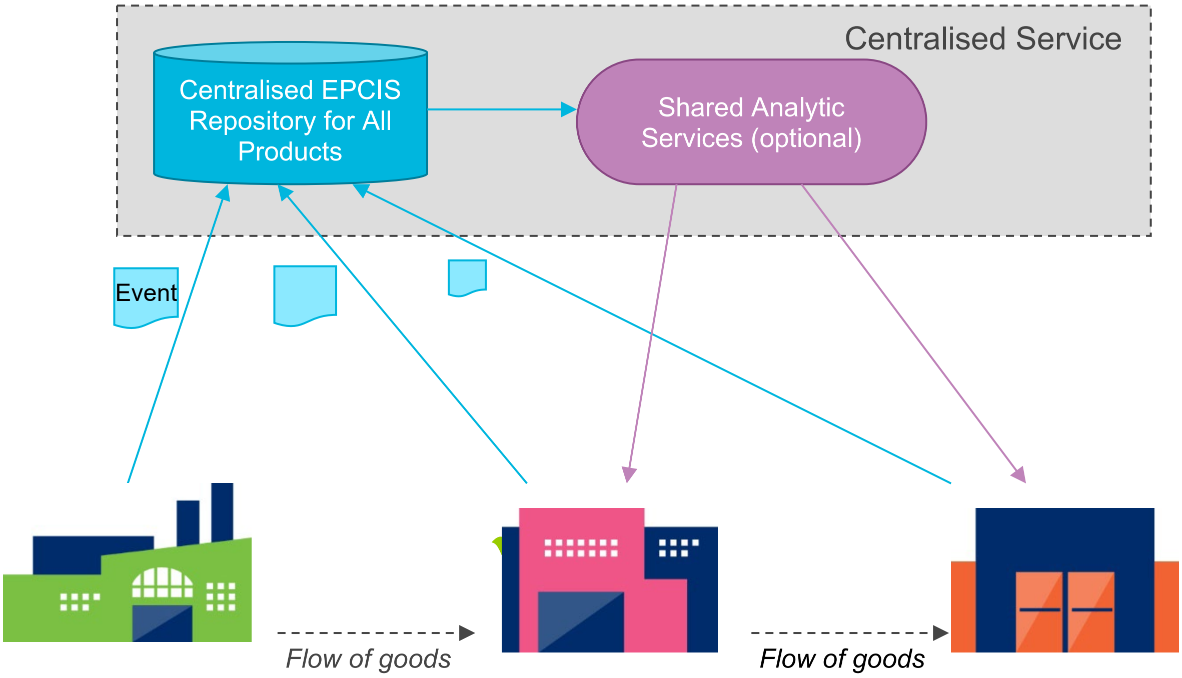
  - When does the producer share its EPCIS events to an outside party?
  - Where does the shared data go?
- Questions for the *consumers* of EPCIS event data:
  - How are events produced by multiple parties gathered together for analysis?
  - Who does the work of gathering events and performing the analysis?

A given party may act as both a producer and a consumer in the context of different business processes.

6.4.1 Centralised Choreography

The simplest choreography model is one in which there is a single EPCIS repository shared by all supply chain parties.

Figure 6-1 Centralised Choreography


Centralised Service
Centralised EPCIS Repository for All Products
Shared Analytic Services (optional)
Event
Flow of goods
Flow of goods

The centralised choreography model has these characteristics:

Table 6-5 Characteristics of Centralised Choreography

Question	Centralised Choreography
When does the producer share its EPCIS events?	As soon as each producer captures its events, or when it ships the product.
Where does the shared data go?	The producer shares its data with the central repository.
How are events gathered for analysis?	All events are present in the central repository, so no additional steps are required to gather events.
Who does the work of gathering events and performing the analysis?	Either the consumer can query the central repository and perform the analysis itself, or the central repository can offer analytic services and do the work on behalf of the consumer.

The centralised approach has the advantage that all events are in one place, simplifying the work of gathering events for analysis. It also provides a natural place to offer shared analytic services.

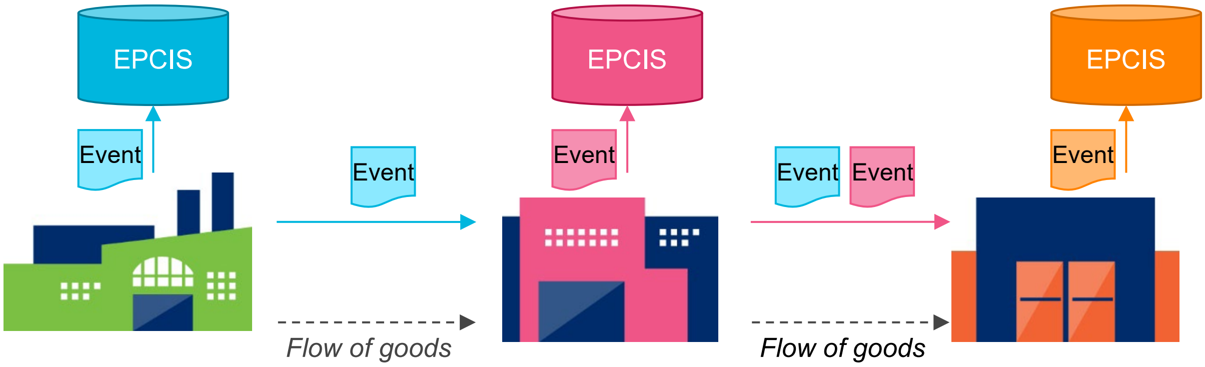
One disadvantage of the centralised approach is that all supply chain parties must agree to use the same repository service. This may not be feasible for a very large supply chain. A variation on the centralised approach is one in which there may be many shared repositories – this is called a *semi-centralised* approach. With more than one repository, the data required for a given analysis may not necessarily all reside in one repository. So the semi-centralised approach requires additional features to mitigate this. Some possibilities include:

- Multiple shared repositories can federate with each other, so that they keep synchronised copies of each other's data or they forward queries to each other as needed.
- If queries are limited to gathering events for a single EPC class (e.g., for a single GTIN), repositories can be segregated on that basis. This requires each EPC class to be associated with a specific repository (typically one nominated by the party that commissions the EPC) and registered in some lookup service that maps an EPC class to specific repository. The Object Name Service (ONS) could be used for that purpose. Each downstream party then uses the lookup service to determine which repository to share its data with.

6.4.2 Distributed Push Choreography

In a Distributed Push Choreography approach, each supply chain party keeps the data it captures in its own EPCIS Repository, and also sends a copy of EPCIS events downstream following the flow of the corresponding physical objects. There are no EPCIS queries involved.

Figure 6-2 Distributed Push Choreography



The distributed push choreography model has these characteristics:

Table 6-6 Characteristics of Distributed Push Choreography

Question	Centralised Choreography
When does the producer share its EPCIS events?	When it ships the physical objects to a downstream party.
Where does the shared data go?	To the downstream party, and to its downstream parties.
How are events gathered for analysis?	Downstream parties receive all of the upstream events, so no additional work is required to gather events.
Who does the work of gathering events and performing the analysis?	The consuming party.

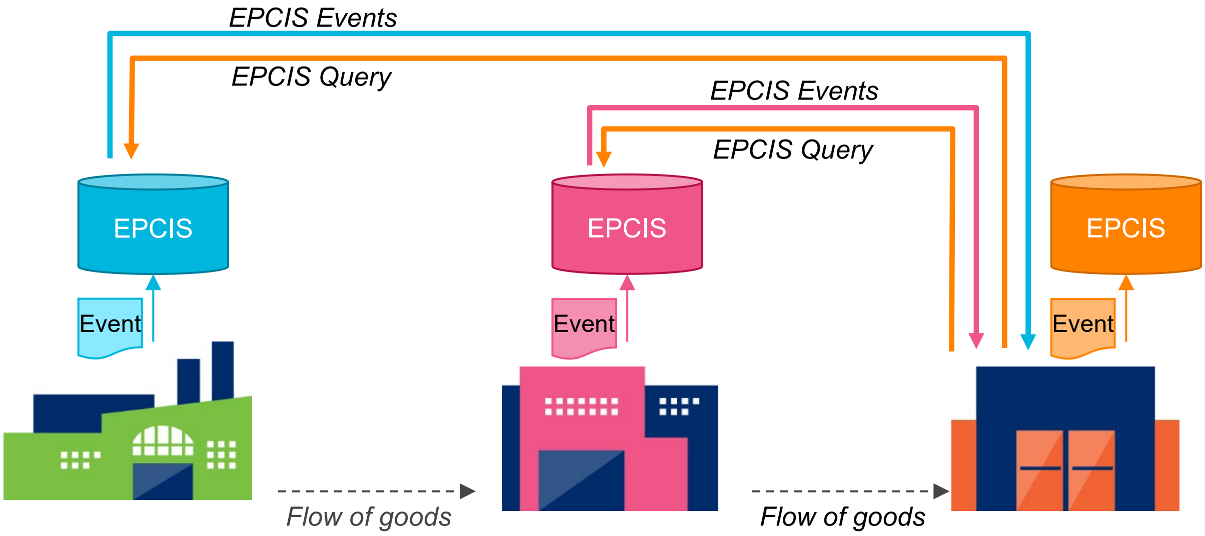
The distributed push approach has the advantage that the consuming party receives the data it needs in advance; there is no need to query for data later. This makes the method robust in that the consuming party does not need to rely on the availability of any party's EPCIS service (or of any shared service). A disadvantage of this approach, however, is that events are communicated whether or not the events are ultimately needed; also, the intermediate parties must relay events even if they have no interested in using them.

As described above, the distributed push approach communicates upstream events to downstream parties. It would be possible for events to be communicated in the opposite direction as well, to provide for downstream events to be received by upstream parties.

6.4.3 Distributed Query Choreography

In a Distributed Query Choreography approach, each supply chain party keeps the data it captures in its own EPCIS Repository. Any party that needs another party's data must query for it.

Figure 6-3 Distributed Query Choreography



The distributed query choreography model has these characteristics:

Table 6-7 Characteristics of Distributed Query Choreography

Question	Centralised Choreography
When does the producer share its EPCIS events?	Only when queried by another party.
Where does the shared data go?	Directly to the party who needs the data.
How are events gathered for analysis?	By making queries to individual parties' EPCIS repositories. This in turn requires some method to discover which EPCIS repositories need to be queried.
Who does the work of gathering events and performing the analysis?	The consuming party.

The distributed query approach has the advantage that each party can keep tight control on their data and only delivers data directly to the party that consumes it (the data does not have to be forwarded through any other party). Also, there is no reliance on any shared service.

A challenge in the distributed query approach is *discovery*: how does the consumer of EPCIS data find the other EPCIS repositories to query? There are several parts to discovery:

- Determining what other parties have (or may have) data that is relevant to the consumer's information need.
- Obtaining a network address of the EPCIS service of each party to be queried.
- Authenticating and establishing trust with each queried party, so that the queried party will be comfortable authorising access to the data the querying party wants. This may be complicated if the querying party does not have a direct business relationship with the queried party; e.g., if they are more than one step removed from each other in the supply chain.

There are many possible approaches to solving the discovery problem, including:

- **Chain of Custody Token**: In this approach, each party in the supply chain sends a short message to the next downstream party in the supply chain containing the network address of its EPCIS service and an authorisation token providing access to EPCIS data pertaining to the specific physical objects being shipped. A party in the middle of the supply chain not only provides its chain of custody token to downstream partners, but also forwards along the tokens it receives from upstream parties. In this way, a downstream party receives tokens that provide access to all upstream parties that have data about the physical objects it receives.

As described, this allows downstream parties to discover upstream data but not the reverse. However, separate tokens could be sent and forwarded upstream to give upstream parties the ability to discover downstream data.

- **Discovery Service**: In this approach, a centralised "discovery service" is maintained which acts as an index for the location of all relevant EPCIS data. When a party captures its own EPCIS data, it sends a message to the discovery service containing the network address of its EPCIS service and identifying the physical objects for which it has data. A consuming party can subsequently query the discovery service to find all of the EPCIS services that have relevant data. The information sent to the discovery service may also include authorisation information so that trust may be established when the consuming party queries the producers.

Note that the discovery problem is similar to the problem of distributing EPCIS events themselves, except that the information distributed is a *pointer* to EPCIS data. From that perspective, a discovery service is a like a centralised model for pointer data, and the chain of custody approach is like the distributed push model for pointer data. A pointer to EPCIS data, however, is less data than EPCIS events themselves, and so there is less data centralised or pushed than there would be in a true centralised or distributed push choreography for EPCIS events.

6.5 Synchronisation of Master Data

Data in the *what* and *where* dimensions of EPCIS events take the form of globally unique identifiers, for example a Serialised Global Trade Item Number (SGTIN) in the *what* dimension or a Global Location Number (GLN) in the *where* dimension. In order to interpret the business meaning of an EPCIS event, a business application typically needs additional descriptive information associated with each identifier. For example, descriptive information for a GTIN might include the name of the product, the brand name, the physical dimensions, and so on. Descriptive information for a GLN might include the street address of the location and its geocoordinates. Such descriptive information is called "master data."

Compared to EPCIS event data, master data is static. Unlike event data, more master data is not created merely because more business is transacted. Master data is not completely static, however: additional master data may be created due to growth, for example when new products are introduced or new physical locations are built. But in general, a given identifier having a single set of associated master data may be mentioned in many different EPCIS events. For this reason, it is desirable to communicate master data in advance, just one time for each distinct identifier, rather than include master data in every EPCIS event.

There are several ways that master data can be communicated from the creator of an identifier to the other parties who may need the master data. These include:

- Using a system designed for the efficient communication of master data. Such systems include:
  - The GS1 Global Data Synchronisation Network (GDSN), for both trade item (GTIN) master data and GLN master data
  - The GS1 GLN Registry federation, for more detailed information about GLNs
- Using the Instance/Lot Master Data (ILMD) feature of EPCIS events to carry master data directly within an event. This applies to master data for specific lots of a GTIN or to specific instances (SGTINs or other instance-level identifiers).
- Using the `VocabularyList` element of the standard EPCIS Header to carry master data in an EPCIS XML document.
- Other means not standardised by GS1.

Of these methods, GDSN and the GLN Registry are fully governed by standards and so offer the greatest degree of interoperability. The ILMD feature, and – for the EPCIS XML binding – the `VocabularyList` element of the EPCIS Header provide a standardised interface for master data, and may be used with the master data attributes defined in the CBV.

6.6 Retrieval of master data using a Web request for the GS1 Digital Link URI

Three of the four existing master data transmission methods for the EPCIS XML binding were seen in 2007 as a migration-path alternative for users who had not yet deployed existing (e.g., GDSN or EANCOM-based) standardised master data exchange methods. These provisions were introduced in the absence of other alternatives, over 10 years before the publication of the GS1 Digital Link standard.

CBV 2.0 allows a constrained set of GS1 Digital Link URIs, corresponding to the EPC schemes specified in GS1's EPC Tag Data Standard (TDS), as identifiers (for class and instance-level objects, locations, parties, transactions and transformations) in EPCIS events, as an alternative to EPC URNs.

Querying master data via a GS1 Digital Link-compliant resolver is considered a forward-looking approach.

A forthcoming GS1 white paper, **"***On-demand access to master data***"** explains how master data could be retrieved using a Web request for the GS1 Digital Link URI for a product, product lot or instance, place or organisation, by making use of terms (properties, classes) already defined within the GS1 Web vocabulary [https://gs1.org/voc].

6.7 Redaction of EPCIS Event Data

A fundamental principle of EPCIS is that the party who captures EPCIS data owns that data, and is in full control of which other parties may receive it. Therefore, merely because one party queries another party for EPCIS events matching some criteria does not mean that the queried party is obligated to respond with all matching events. Instead, the queried party may choose to restrict what data the querying party receives based on business rules. This is termed "redaction."

In general, an EPCIS service that is sending data to another party, whether in response to a query or due to some other trigger, may consider the identity of the receiving party and apply business rules to redact the data. The following possibilities for redaction are paraphrased from the EPCIS 1.1 standard, section 8.2.2

- The service may refuse to honour the request altogether, by responding with a Security Exception
- The service may respond with less data than requested. For example, if a querying party presents a query requesting all Object Event instances within a specified time interval, the service knows of 100 matching events, the service may choose to respond with fewer than 100 events (e.g., returning only those events whose EPCs are SGTINs with a company prefix known to be assigned to the querying party).
- The service may respond with coarser grained information. In particular, when the response to a query includes a location identifier the service may substitute an aggregate location in place of a primitive location (for example, a site-level GLN instead of the SGLN of a particular loading dock).
- The service may hide information. For example, if a querying party presents a query requesting Object Event instances, the service may choose to delete the `bizTransactionList` fields in its response. The information returned, however, shall always be well-formed EPCIS events consistent with this specification and industry guidelines. For example, given an `AggregationEvent` with action equal to `ADD`, an attempt to hide the `parentID` field would result in a non-well-formed event, because `parentID` is required when the action is `ADD`; in this instance, therefore, either the `parentID` would have to be included or the entire event would have to be withheld.
- The service may limit the scope of the query to data that was originally captured by a particular client identity. This allows a single EPCIS service to be partitioned for use by groups of unrelated users whose data should be kept separate (a so-called "multi-tenant" implementation).

An EPCIS implementation is free to determine which if any of these actions to take in processing any query, using any means it chooses. The specification of authorisation rules is outside the scope of the EPCIS standard: the EPCIS standard does not take a position as to how authorisation decisions are taken. Particular implementations of EPCIS may have arbitrarily complex business rules for authorisation.

7 Data Validation and System Interoperability

7.1 Validation of EPCIS events

The functioning of EPCIS-based visibility systems greatly depends on the data quality of EPCIS events. For this purpose, organisations should apply validation mechanisms. These include technical, content, and integrity validation:

- **Technical validation** implies that the EPCIS events conform to the current EPCIS standard from a technical perspective. In other words, events are transmitted in XML format according to the XML schema (XSD) specified in the EPCIS 1.1 standard. For specific use cases involving user or vendor extensions, best practise is to build a XSD covering the extra namespaces and XML elements required for these use cases and consider it for technical validation as well.
- **Content validation** requires verification that discrete events make sense from a business perspective. For example, if the process flow in a specific use cases specifies that a pallet packing event should include an SSCC and a quantity of cases described with an LGTIN, then content validation would confirm that the packing event has that structure and not some other structure (which may be syntactically valid, but not appropriate for the specific use case). Additionally, a capture application of an EPCIS might perform semantic checks like date validation, for example to confirm the value of `eventTime` is not in the future.
- **Integrity validation** requires that the visibility system operates end-to-end in the way described by the process map and achieves the desired business results. For example, a requirement could be that it is possible to trace back an item from the goods issue to the receiving process within a location.

Both technical and content validation can usually be accomplished at the very moment EPCIS events are submitted via the EPCIS capture interface. Depending on how important data quality is, an EPCIS repository or capturing application may only accept incoming EPCIS events if they fulfil a predefined set of technical and content validation criteria (and reject them otherwise). Alternatively, incoming events could be accepted as long as they have passed the technical validation, with warnings generated if the content validation has failed.

Integrity validation however can only be accomplished retrospectively, that is, after all events comprising an end-to-end-process have been captured. A business application which consumes visibility event data may apply appropriate rules to deal with invalid event sequences. Amongst other things, it may trigger an alert if a mandatory event to a specific business process does not exists, or it may disregard events which are obvious duplicates or which have no significance.

7.2 Certification program

The EPCglobal Software Certification Program is a standards-based compliance testing program, developed by the EPCglobal community to provide a neutral and authoritative source for testing EPC/RFID software products and providing information regarding certified products and the vendors who develop them.

7.3 Requirements of program certification

EPC Information Services (EPCIS) 1.0 Specification Conformance Requirements are published at http://www.gs1.org/gsmp/kc/epcglobal/epcis.

7.4 Non-normative tools

GS1 maintains a set of non-normative tools at https://ref.gs1.org/tools/ , with EPCIS-specific tools linked from https://ref.gs1.org/tools/epcis/ .

8 References

[CBV] GS1, "Core Business Vocabulary (CBV) Standard, Release 2.0," GS1 Standard, June 2022, https://ref.gs1.org/standards/cbv/.

[CEFACT20] United Nations Economic Commission for Europe, "Recommendation 20: Codes for Units of Measure Used in International Trade," http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev7e_2010.zip.

[CURIE] Compact URI Expressions, https://www.w3.org/TR/curie/

[EPCIS] GS1, "EPCIS Standard, Release 2.0," GS1 Standard, June 2022, https://ref.gs1.org/standards/epcis/.

[GS1Arch] "The GS1 System Architecture," GS1 technical document,

http://www.gs1.org/docs/gsmp/architecture/GS1_System_Architecture.pdf

[GS1DL] GS1 Digital Link Standard: URI Syntax, https://www.gs1.org/standards/gs1-digital-link

[GenSpecs] GS1, "GS1 General Specifications," GS1 Standard, http://www.gs1.org/docs/barcodes/GS1_General_Specifications.pdf.

[JSON] JavaScript Object Notation, https://www.json.org

[JSON-LD] JSON for Linked Data v1.1, https://json-ld.org, *https://www.w3.org/TR/json-ld11/*

[TDS] GS1, "EPC Tag Data Standard (TDS), Release 2.0," GS1 Standard, August 2022, https://ref.gs1.org/standards/tds/2.0.0/.

[XML] https://www.w3.org/XML/

9 Appendix: XML and JSON-LD Examples

Sample XML and JSON-LD for EPCIS events can be found in standalone files at https://ref.gs1.org/docs/epcis/examples/

XML and JSON-LD xamples at https://ref.gs1.org/docs/epcis/examples/ referenced in this Guideline have a file naming convention of `epcis_guideline_example_[guideline section]-[subsection example number].`

In many of the tabular examples, one or more EPCIS dimensions are omitted for clarity, and placeholders like "GTIN X" are used instead of actual identifiers. In the standalone XML and JSON-LD examples, all such omitted details are included using sample values.