別添

EPCIS データ連携
ガイドライン

暫定α版 発行

東芝テック株式会社

目次

1. 概要

本資料では、EPCIS データ連携ミドルウェアついて説明する。EPCIS は分散型データシステムでありながら、日本のように高度な製配販の物流モデルの中で実装された例としては報告がない。そのため EPCIS が、日本型の製配販の物流モデルに耐えうるものであるか、また、そのために何が必要かを実証する試験をおこなった。その結果、本書で紹介する EPCIS データ連携ミドルウェアがあれば、日本でも十分実用に耐えうることが分かり、そのために EPCIS の仕様を追加・変更する必要がないことが判明した。また、EPCIS に用意されているコア・ビジネスボキャブラリを使えば、実証試験において、ボキャブラリを追加する必要がほぼないことも分かった。

1.1. システムアーキテクチャ

EPCIS のシステムアーキテクチャを次の図に示す。

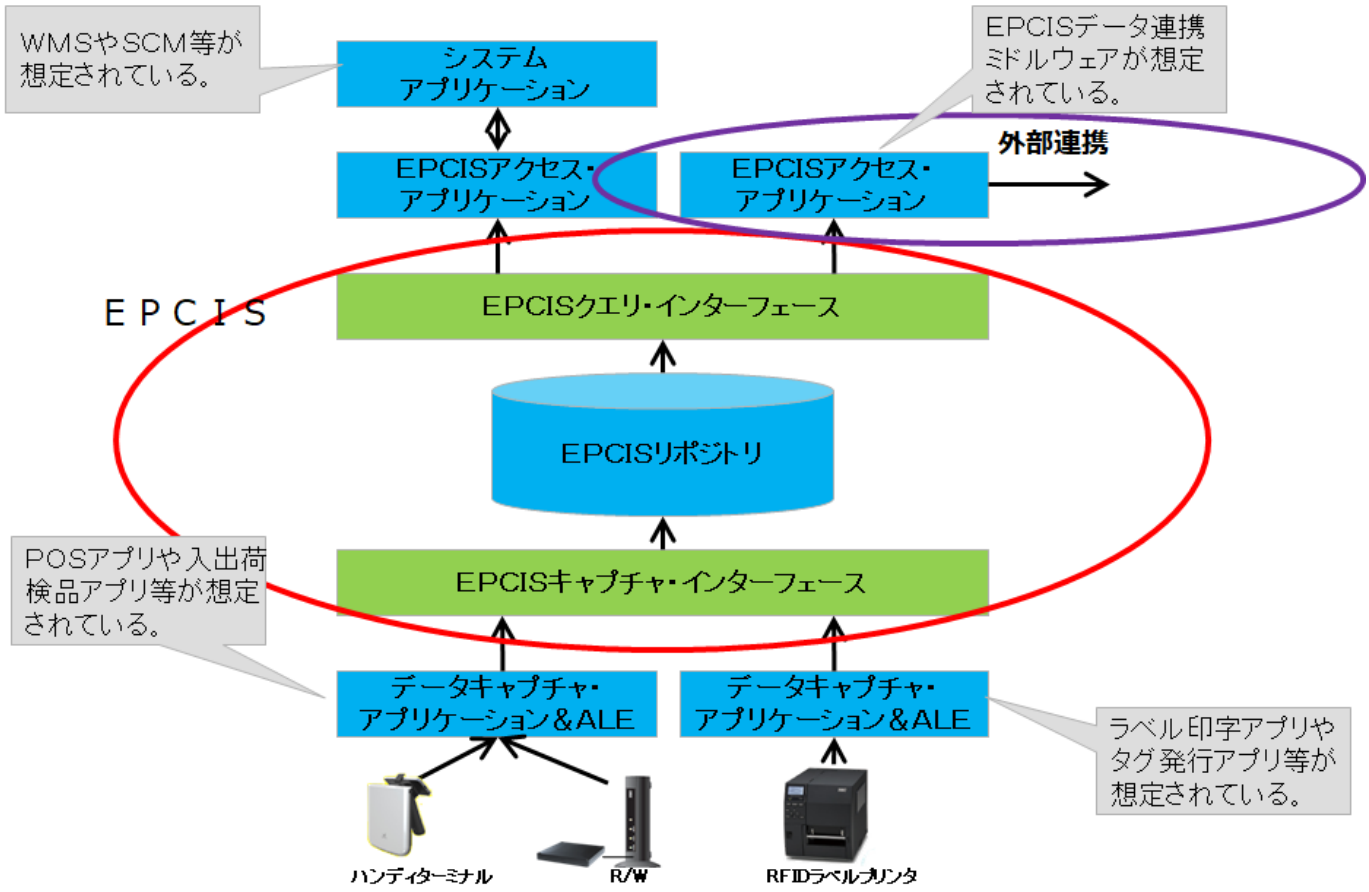

図 1-1 EPCIS のシステムアーキテクチャ

2. EPCIS データ連携ミドルウェアの概要

EPCIS データ連携を実現する方法は、EPCIS 及び CBV 導入ガイドラインに示されるようにいくつかの方法がある。本書では、その中の「分散型プッシュ・コレオグラフィ」を実現する方法について述べる。本 EPCIS データ連携ミドルウェアを使うと、取引先の EPCIS に対して、自分の持っている EPCIS から取引先に対して、開示していいデータだけを送ることができる。そのためには、EPCIS イベントデータの source,destination データを必須データにする必要があるが、それだけで、EPCIS に備わっている EPCIS クエリコールバック・インタフェースを変更する必要がなく行える優位性がある。

2.1. EPCIS と EPCIS データ連携ミドルウェアの関係

EPCIS と EPCIS データ連携ミドルウェアの関係を次の図に示す。

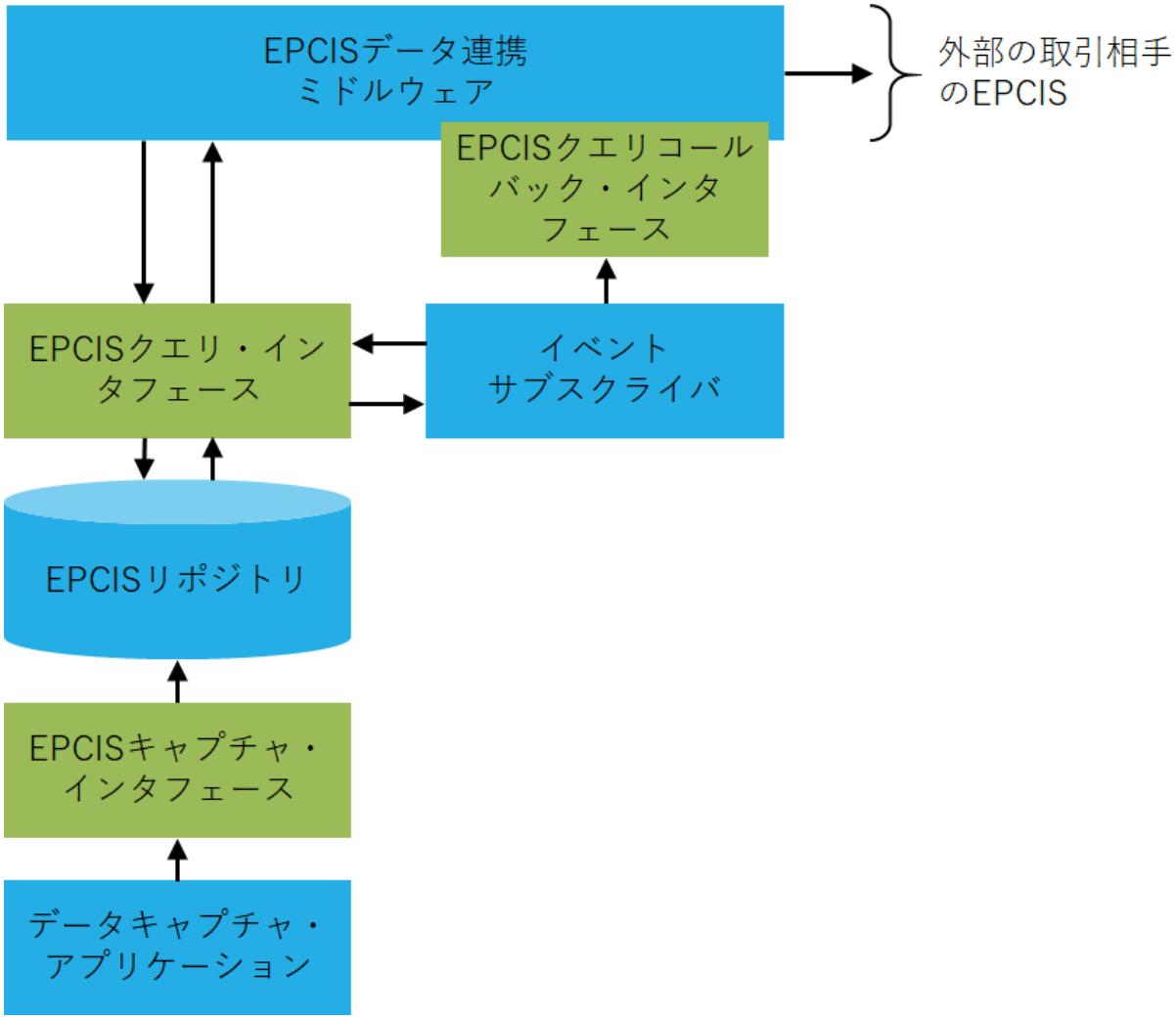

図 2-1 EPCIS と EPCIS データ連携ミドルウェアの関係

EPCIS データ連携ミドルウェアは EPCIS クエリコールバック・インタフェースを備えた EPCIS クエリコールバック・アプリケーションであるとともにデータキャプチャ・アプリケーションでもある。共有の対象となるイベントがデータキャプチャ・アプリケーションにより EPCIS リポジトリへキャプチャされると、イベントサブスクライバから、EPCIS クエリコール

バック・インタフェースを通して EPCIS データ連携ミドルウェアに当該イベントが通知され、EPCIS データ連携ミドルウェアは通知されたイベントの内容に応じて、EPCIS クエリ・インタフェースを介して EPCIS リポジトリから必要なデータを取得、共有データを作成し、外部の取引相手の EPCIS へイベントデータを送信する。

2.2. EPCIS 間のデータ共有

EPCIS 間でデータの共有を行う手順を次の図に示す。

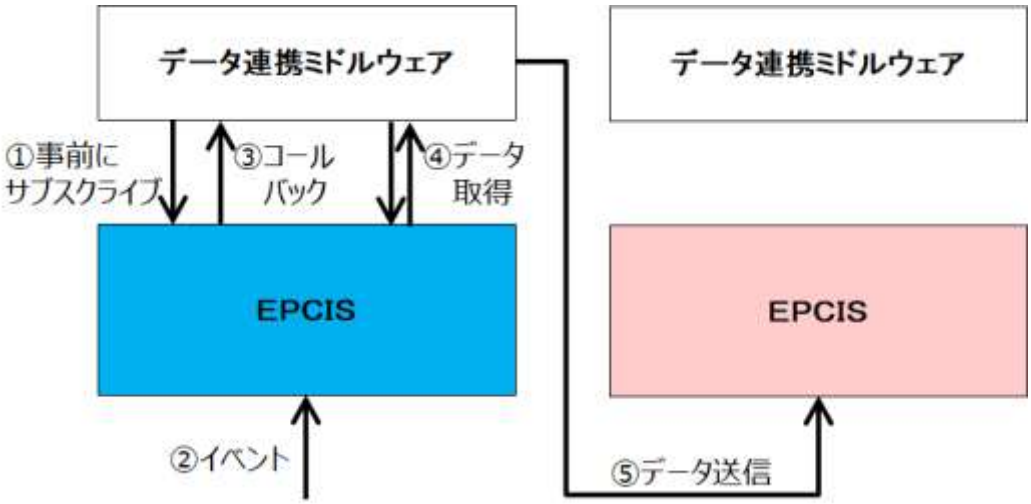

図 2-2 EPCIS 間のデータ共有手順

データ共有は次の流れにより実行される。

① EPCIS データ連携ミドルウェアへ連携対象のイベントキャプチャを通知するサブスクリプションを登録する

② EPCIS に連携対象のイベントがキャプチャされる

③ EPCIS が連携対象のイベントがキャプチャされたことを EPCIS データ連携ミドルウェアに通知する

④ EPCIS データ連携ミドルウェアは通知された連携対象イベントの情報から関連イベントの収集、フィルタリング、マスタデータの取得等を実行し、相手先の EPCIS へ共有する情報を取得する

⑤ 連携対象イベントの宛先情報(SGLN)から相手先の EPCIS のアドレスを特定し、その EPCIS に対して連携対象のイベント及び④で取得した情報を送信する。

次の図に製造事業者、卸事業者、小売事業者の三者間で入出荷および販売情報を共有する場合のデータ共有フローを示す。

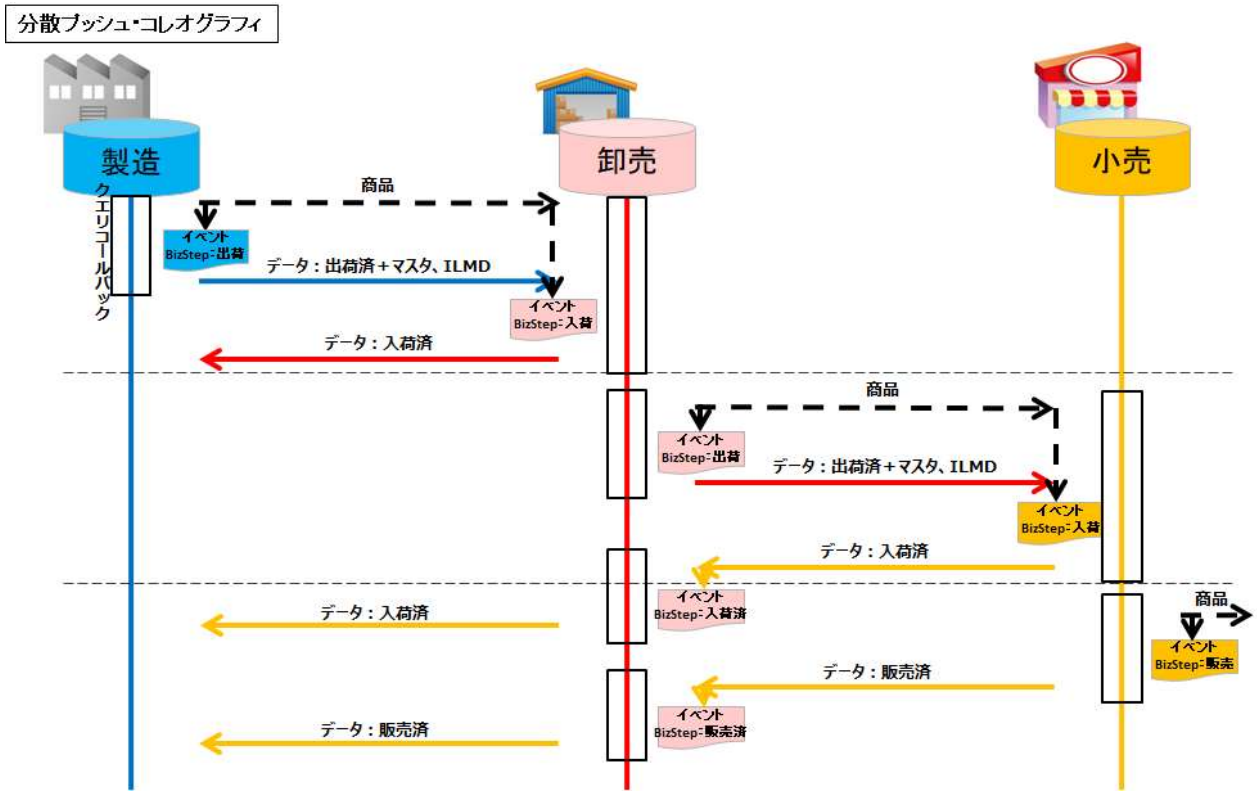

図 2-3 入出荷および販売情報のデータ共有フロー

この図にて想定されるシナリオは以下の通りである。

	業務/処理	事業者	内容
1	タグ発行	製造	商品製造時にRFIDタグを発行すると共に、消費期限などのロット情報を登録する。
2	出荷処理	製造	商品出荷時に宛先を指定して、RFIDタグ（SGTIN）を読み取る。
3	データ連携	製造	EPCISから商品の宛先の卸事業者のEPCISに対して、対象商品のロット情報、出荷データ等を送信する。
4	入荷処理	卸	製造事業者からの出荷データを使用してRFIDタグ（SGTIN）を読み取り、入荷処理を実施。
5	データ連携	卸	入荷データをデータ連携先製造事業者のEPCISに返送。
6	出荷処理	卸	商品出荷時に小売事業者の宛先を指定して、RFIDタグ（SGTIN）を読み取る。
7	データ連携	卸	EPCISから商品出荷先小売事業者のEPCISに対して、対象商品のロット情報、卸からの出荷データ等を送信。
8	入荷処理	小売	卸事業者からの出荷データを使用してRFIDタグ（SGTIN）を読み取り、入荷処理を実施。
9	データ連携	小売	入荷データをデータ連携先卸事業者のEPCISに返送。
10	データ連携	卸	EPCIS にて、小売事業者から返送された入荷データをデータ連携先製造事業者のEPCISに返送。

11	販売処理	小売	販売時に RFID タグを読み取り会計を実施。
12	データ連携	小売	対象商品の販売イベントをデータ連携先卸事業者の EPCIS に送信。
13	データ連携	卸	小売事業者から返送された販売イベントをデータ連携先製造事業者の EPCIS に返送。

3. EPCIS データの共有

3.1. イベントのサブスクライブ

EPCIS データの共有を行うためには、EPCIS へ EPCIS データの共有のトリガとなるイベントに対するサブスクリプションの登録を行う必要がある。サブスクライブ対象となるイベントは外部の取引相手と EPCIS データの共有を実行する必要のあるタイミングで発生するイベントで、どのイベント発生のタイミングで共有するかについてはサプライチェーンに関わる事業者間の合意により取り決める必要がある。一般的には、外部の取引相手とのやり取りが発生したタイミングでデータを共有する。例えば、出荷、入荷、販売などである。サブスクリプションの設定例を次の表に示す。

表 3-1 出荷イベントサブスクリプション設定例

項目	設定値
subscriptionID	shipping001
queryName	SimpleEventQuery
queryParams	“eventType”, { “ObjectEvent” }
	“EQ_bizStep”, { “urn:epcglobal:cbv:bizstep:shipping” }
dest	http://www.example.co.jp/epcis/queryCallback
reportIfEmpty	false
schedule	second “0,30”

表 3-2 入荷イベントサブスクリプション設定例

項目	設定値
subscriptionID	receiving001
queryName	SimpleEventQuery
queryParams	“eventType”, { “ObjectEvent” }
	“EQ_bizStep”, { “urn:epcglobal:cbv:bizstep:receiving” }
dest	http://www.example.co.jp/epcis/queryCallback
reportIfEmpty	false
schedule	second “0,30”

表 3-3 販売イベントサブスクリプション設定例

項目	設定値
subscriptionID	retail_selling001
queryName	SimpleEventQuery
queryParams	“eventType”, { “ObjectEvent” }
	“EQ_bizStep”, { “urn:epcglobal:cbv:bizstep:retail_selling” }
dest	http://www.example.co.jp/epcis/queryCallback
reportIfEmpty	false
schedule	second “0,30”

3.2. 関連イベントの収集と共有

EPCIS データ連携ミドルウェアは、イベントサブスクライバからコールバックされたイベントの内容をもとにそのイベントを必要とする外部の取引相手に EPCIS クエリ・インタフェースを介して自社の EPCIS リポジトリより検索し、収集し渡す。関連イベントとはイベントに含まれている EPC の内、一つ以上を含むイベントのことである。次の表に製造事業者、卸事業者、小売事業者の三事業者間でサプライチェーンを物品が通過する際に発生するイベントの例を示す。

表 3-4 生産から販売までのサプライチェーン通過中に発生するイベントの例

		製造		卸		小売	
		V1	V2	V3	V4	V5	V6
		タグ発行	出荷	入荷	出荷	入荷	販売
	eventType	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent
	action	ADD	OBSERVE	OBSERVE	OBSERVE	OBSERVE	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1
Why	bizStep	commissioning	shipping	receiving	shipping	receiving	retail_selling
	disposition	active	in_transit	in_progress	in_transit	in_progress	retail_sold
	source		製造の SGLN	製造の SGLN	卸の SGLN	卸の SGLN	小売の SGLN
	destination		卸の SGLN	卸の SGLN	小売の SGLN	小売の SGLN	

製造事業者、卸事業者、小売事業者の間でこれらのデータを共有することで、物品のサプライチェーン通過中の状況を把握することができる。共有対象とするイベントはサプライチェーンに関わる事業者間の合意により取り決める必要があるが、表で示す例においては例えば、以下のような共有を実施する。

- V1 :なし
- V2 :製造事業者から卸事業者へ V1、V2 を共有
- V3 :卸事業者から製造事業者へ V3 を共有
- V4 :卸事業者から小売事業者へ V1、V2、V3、V4 を共有
  卸事業者から製造事業者へ V4 を共有
- V5 :小売事業者から卸事業者へ V5 を共有
  卸事業者から製造事業者へ V5 を共有
- V6 :小売事業者から卸事業者へ V6 を共有
  卸事業者から製造事業者へ V6 を共有

これにより、各事業者の EPCIS へ表 3-4 に示す EPCIS データが共有され、各事業者は自社の EPCIS のみを参照することで各種データ分析、把握が可能となる。

3.2.1. AggregationEvent による集約と集約の解除

AggregationEvent は parentID により指定される物品に childEPCs で指定される一つ以上の物品を集約、または集約解除する。例えば、箱に物品が梱包される、箱から物品が取り出される、パレットに物品が積み込まれる、パレットから物品が積降ろされるなどである。このとき、次の表の V4 に示すように出荷時には集約された ID のみが通知されることが考えられる。共有先へは箱やパレットの情報だけでなく、梱包されているもの、積み込まれているものの情報も共有する必要があるため、関連イベント収集の際に AggregationEvent が検出された場合にはその集約および集約解除の情報を追跡する必要がある。

表 3-5 AggregationEvent による集約・集約解除の例

		製造			
		V1	V2	V3	V4
		タグ発行	梱包	開梱	出荷
	eventType	ObjectEvent	AggregationEvent	AggregationEvent	ObjectEvent
	action	ADD	ADD	DELETE	OBSERVE
What	parentID		sscc1	sscc1	
	epcList childEPCs	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3	sgtin2	sscc1
Why	bizStep	commissioning	packing	unpacking	shipping
	disposition	active	in_progress	in_progress	in_transit
	source				製造の SGLN
	destination				卸の SGLN

この場合の関連イベントの収集手順は例えば以下のようになる。

1. 表 3-5 の V4 の情報から sscc1 を検索し、V2,V3 を得る
2. AggregationEvent が含まれるので集約の関係を調べる
3. 時系列の昇順に並べ時間が古い順に V2,V3 とする
4. V2 の情報から sscc1 の梱包リストに sgtin1,sgtin2,sgtin3 を追加
5. V3 の情報から sscc1 の梱包リストから sgtin2 を削除
6. 4，5の手順により得られた梱包リストから sgtin1,sgtin3 を検索し、V1 を得る

最終的に V4 に関連するイベントは V1,V2,V3 となる。

3.3. データのフィルタリング

EPCIS リポジトリに格納されているデータを未加工のまま、外部の取引相手の EPCIS へ共有した場合、物品の情報が無関係の事業者へ流れてしまう可能性がある。これを防ぐため、EPCIS データ連携ミドルウェアはイベントに含まれる情報のフィルタリングを行ってもよい。次の表にデータのフィルタリングを行わずに製造事業者から卸事業者へ出荷時の EPCIS データを共有する例を示す。

表 3-6 製造事業者から卸事業者への出荷時 EPCIS データ共有(フィルタリングなし)

		製造	
		V1	V2
		タグ発行	出荷
	eventType	ObjectEvent	ObjectEvent
	action	ADD	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3
Why	bizStep	commissioning	shipping
	disposition	active	in_transit
	source		製造の SGLN
	destination		卸の SGLN

共有

卸	
V1	V2
タグ発行	出荷
ObjectEvent	ObjectEvent
ADD	OBSERVE
sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3
commissioning	shipping
active	in_transit
	製造の SGLN
	卸の SGLN

このように、フィルタリングを行わない場合、V2 にて製造事業者から卸事業者へ出荷した物品は(sgtin1, sgtin2, sgtin3)の三点であるが、関連するイベントとして合わせて共有された V1 について、出荷していない物品である sgtin4 の情報が共有されてしまう。このような場合、次の表に示すように卸事業者にとって無関係のデータである sgtin4 を EPCIS データから取り除いた状態で共有することが望ましい。

表 3-7 製造事業者から卸事業者への出荷時 EPCIS データ共有(フィルタリングあり)

		製造			卸	
		V1	V2		V1	V2
		タグ発行	出荷		タグ発行	出荷
	eventType	ObjectEvent	ObjectEvent		ObjectEvent	ObjectEvent
	action	ADD	OBSERVE		ADD	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3	共有	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3
Why	bizStep	commissioning	shipping		commissioning	shipping
	disposition	active	in_transit		active	in_transit
	source		製造の SGLN			製造の SGLN
	destination		卸の SGLN			卸の SGLN

次に表 3-4 で示したサプライチェーンで発生するイベントの例についてフィルタリングを実施し、無関係のデータを含まないように共有された製造事業者、卸事業者、小売事業者のそれぞれの EPCIS データの例を示す。

表 3-8 製造事業者のデータ共有例

		製造		卸		小売	
		V1	V2	V3	V4	V5	V6
		タグ発行	出荷	入荷	出荷	入荷	販売
	eventType	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent
	action	ADD	OBSERVE	OBSERVE	OBSERVE	OBSERVE	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1
Why	bizStep	commissioning	shipping	receiving	shipping	receiving	retail_selling
	disposition	active	in_transit	in_progress	in_transit	in_progress	retail_sold
	source		製造の SGLN	製造の SGLN	卸の SGLN	卸の SGLN	小売の SGLN
	destination		卸の SGLN	卸の SGLN	小売の SGLN	小売の SGLN	

表 3-9 卸事業者のデータ共有例

		製造		卸		小売	
		V1	V2	V3	V4	V5	V6
		タグ発行	出荷	入荷	出荷	入荷	販売
	eventType	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent
	action	ADD	OBSERVE	OBSERVE	OBSERVE	OBSERVE	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1
Why	bizStep	commissioning	shipping	receiving	shipping	receiving	retail_selling
	disposition	active	in_transit	in_progress	in_transit	in_progress	retail_sold
	source		製造の SGLN	製造の SGLN	卸の SGLN	卸の SGLN	小売の SGLN
	destination		卸の SGLN	卸の SGLN	小売の SGLN	小売の SGLN	

表 3-10 小売事業者のデータ共有例

		製造		卸		小売	
		V1	V2	V3	V4	V5	V6
		タグ発行	出荷	入荷	出荷	入荷	販売
	eventType	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent	ObjectEvent
	action	ADD	OBSERVE	OBSERVE	OBSERVE	OBSERVE	OBSERVE
What	epcList	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1
Why	bizStep	commissioning	shipping	receiving	shipping	receiving	retail_selling
	disposition	active	in_transit	in_progress	in_transit	in_progress	retail_sold
	source		製造の SGLN	製造の SGLN	卸の SGLN	卸の SGLN	小売の SGLN
	destination		卸の SGLN	卸の SGLN	小売の SGLN	小売の SGLN	

また、各データの内容についてのフィルタリングのほか、物品の追跡に不要なイベントのフィルタリング処理を実施してもよい。

例えば、次のようなイベントは不要となる可能性がある。

1. 製造事業者が工場内での物品の位置把握のためにゲートなどを設置して、キャプチャしたイベント
   製造事業者以外は製造事業者の工場内での詳細な物品の動きについての情報を必要とすることがないと想定される。そのため、次の表に示すように、製造事業者の V1 と V2 の間に発生したゲート通過イベントは卸事業者に共有する意味がなく、イベント自体を共有しないとすることができる。

表 3-11 特定事業者内でのみ関心のあるイベントの除外

		製造				卸	
		V1		V2		V1	V2
		タグ発行	ゲート通過	出荷	共有	タグ発行	出荷
	eventType	ObjectEvent	ObjectEvent	ObjectEvent		ObjectEvent	ObjectEvent
	action	ADD	OBSERVE	OBSERVE		ADD	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3 sgtin4	sgtin1 sgtin2 sgtin3		sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3
Why	bizStep	commissioning		shipping		commissioning	shipping
	disposition	active	in_progress	in_transit		active	in_transit
	source			製造の SGLN			製造の SGLN
	destination			卸の SGLN			卸の SGLN

2. 直接の取引先以外の出荷イベント
   直接の取引先では入荷検品のための情報として出荷イベントが必要となるが、入荷イベントのみが存在すれば物品の追跡が可能であるため、間接的な出荷イベントは必要とすることがないと想定される。
   小売事業者において、次の表に示すように製造事業者と卸事業者間の出荷イベントが共有されていない場合においても、V4 に含まれる epcList の情報(sgtin1,sgtin2)から V1 のイベントまで到達が可能である。

表 3-12 間接取引先の出荷イベントの除外

		卸				小売		
		V1	V2	V3	V4	V1	V3	V4
		タグ発行	出荷	入荷	出荷	タグ発行	入荷	出荷
	eventType	Object Event	Object Event	Object Event	Object Event	Object Event	Object Event	Object Event
	action	ADD	OBSERVE	OBSERVE	OBSERVE	ADD	OBSERVE	OBSERVE
What	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1 sgtin2	sgtin1 sgtin2
Why	bizStep	commissioning	shipping	receiving	shipping	commissioning	receiving	shipping
	disposition	active	in_transit	in_progress	in_transit	active	in_progress	in_transit
	source		製造の SGLN	製造の SGLN	卸の SGLN		製造の SGLN	卸の SGLN
	destination		卸の SGLN	卸の SGLN	小売の SGLN		卸の SGLN	小売の SGLN

3.3.1. AggregationEvent の考慮

データ共有対象となる関連イベントに AggregationEvent が含まれる場合、集約の状態を確認し、集約に無関係の情報を除外してもよい。例えば、表 3-5 で示したイベントを卸事業者へ共有する場合、次の表に示すように sscc1 に含まれていない sgtin2 は除外されることが望ましい。

表 3-13 AggregationEvent を含む関連イベントを共有する場合

		卸		
		V1	V2	V4
		タグ発行	梱包	出荷
	eventType	ObjectEvent	AggregationEvent	ObjectEvent
	action	ADD	ADD	OBSERVE
What	parentID		sscc1	
	epcList	sgtin1 sgtin3	sgtin1 sgtin3	sscc1
Why	bizStep	commissioning	packing	shipping
	disposition	active	in_progress	in_transit
	source			製造の SGLN
	destination			卸の SGLN

このようにデータ共有を行うために、表 3-5 の説明手順に加えて、sscc1 の梱包リスト(sgtin1,sgtin3)を用いて、V1,V2,V3 から無関係の EPC である sgtin2 および sgtin4 を除外すると次の表に示す状態となる。

表 3-14 sscc1 に含まれない sgtin の除外

		製造			
		V1	V2	V3	V4
		タグ発行	梱包	開梱	出荷
	eventType	ObjectEvent	AggregationEvent	AggregationEvent	ObjectEvent
	action	ADD	ADD	DELETE	OBSERVE
What	parentID		sscc1	sscc1	
	epcList childEPCs	sgtin1 sgtin3	sgtin1 sgtin3		sscc1
Why	bizStep	commissioning	packing	unpacking	shipping
	disposition	active	in_progress	in_progress	in_transit
	source				製造の SGLN
	destination				卸の SGLN

このとき、無関係の EPC の除外により epcList/childEPCs が空になったものはイベントとして無効であるため、イベント自体を除外する。これにより、表 3-13 に示した卸に共有するイベントを導出することが可能である。

3.3.2. 再利用資産の循環

パレットや折り畳みコンテナ、カゴ台車などの再利用資産を利用して入出荷を行う場合、次の図に示すように出荷元事業者と出荷先事業者の間でやり取りが行われる。

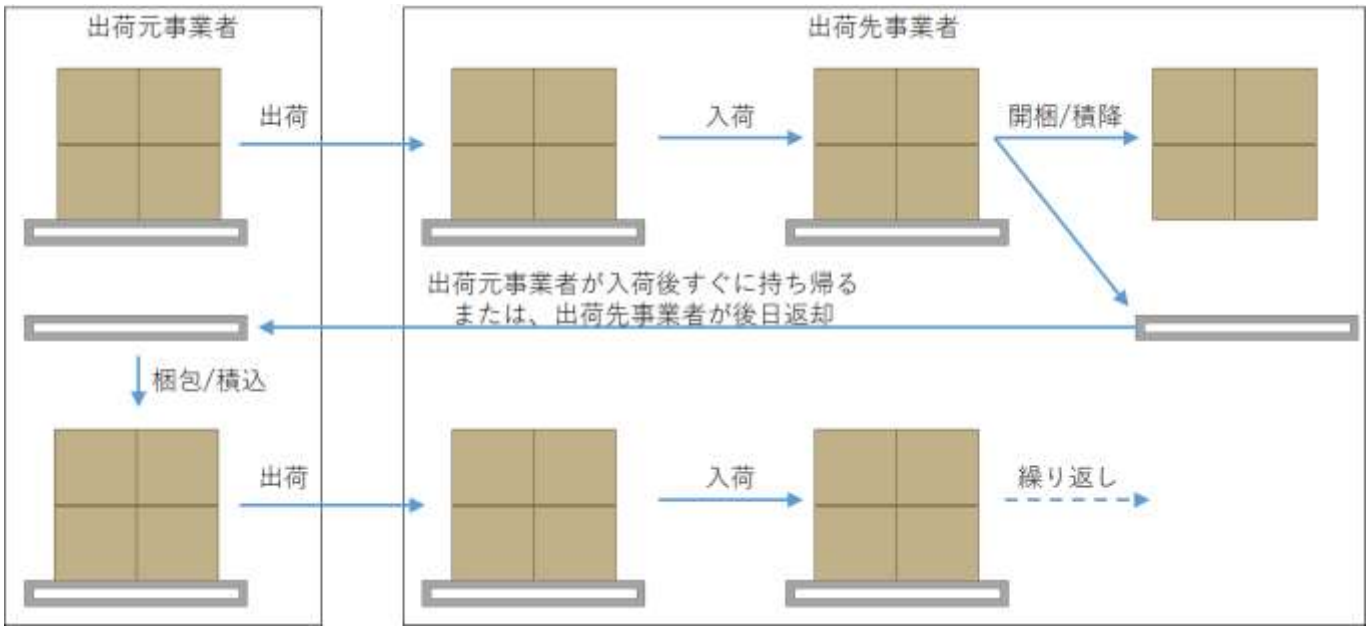

図 3-1 再利用資産の循環

再利用資産が初めて使用される場合、イベントは次のように発生する。

表 3-15 再利用資産を使用した入出荷(1 巡目)

		出荷元事業者			出荷先事業者	
		V1	V2	V3	V4	V5
		タグ貼付	梱包/積込	出荷	入荷	開梱/積降
	eventType	ObjectEvent	Aggregation Event	ObjectEvent	ObjectEvent	Aggregation Event
	action	ADD	ADD	OBSERVE	OBSERVE	DELETE
What	parentID		grai1			grai1
	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	grai1	grai1	sgtin1 sgtin2 sgtin3
Why	bizStep	commissioning	packing/ loading	shipping	receiving	unpacking/ unloading
	disposition	active	in_progress	in_transit	in_progress	
	source			出荷元の SGLN	出荷元の SGLN	
	destination			出荷先の SGLN	出荷先の SGLN	

次に 1 巡目で利用された再利用資産が持ち帰られるか、または返却されたのち、再度利用される場合、表 3-15 に続いて次の表に示すように V6,V7,V8 のイベントが発生する。

表 3-16 再利用資産を利用した入出荷(2 巡目)

		出荷元事業者			出荷先事業者		出荷元事業者		
		V1	V2	V3	V4	V5	V6	V7	V8
		タグ貼付	梱包/積込	出荷	入荷	開梱/積降	タグ貼付	梱包/積込	出荷
	eventType	ObjectEvent	Aggregation Event	ObjectEvent	ObjectEvent	Aggregation Event	ObjectEvent	Aggregation Event	ObjectEvent
	action	ADD	ADD	OBSERVE	OBSERVE	DELETE	ADD	ADD	OBSERVE
What	parentID		grai1			grai1		grai1	
	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	grai1	grai1	sgtin1 sgtin2 sgtin3	sgtin4 sgtin5 sgtin6	sgtin4 sgtin5 sgtin6	grai1
Why	bizStep	commissioning	packing/ loading	shipping	receiving	unpacking/ unloading	commissioning	packing/ loading	shipping
	disposition	active	in_progress	in_transit	in_progress		active	in_progress	in_transit
	source			出荷元の SGLN	出荷元の SGLN				出荷元の SGLN
	destination			出荷先の SGLN	出荷先の SGLN				出荷先の SGLN

この表の V8 時点で出荷元事業者の EPCIS リポジトリに格納されていると想定されるイベントは次の表に示すとおりである。このとき、V8 の出荷イベント発生時に出荷元事業者から出荷先事業者へ共有すべきイベントは V6,V7,V8 であるが、出荷元事業者にて V8 に含まれる EPC である grai1 で EPCIS リポジトリを単純に検索すると、V2,V3,V4,V7 が結果として得られる。V2,V3,V4 を今回の出荷とは無関係と識別するためには V8 時点での grai1 の最新の集約状態を調べる必要があるが、表 3-17 に示すようなイベントの共有状態では V5 のイベントが欠落しているために、grai1 に V8 時点で集約されている物品を洗い出すことはできない。(ここで、V5 のイベントは出荷先事業者が更に下流の事業者へ出荷するなどしていれば共有されていることもあるが、それよりも早く出荷元事業者にて再利用資産が再利用された場合には共有されていない状態となると考えられる。)

表 3-17 V8 イベント発生時点で出荷元事業者の EPCIS リポジトリに格納されていると想定されるイベント

		出荷元事業者						
		V1	V2	V3	V4	V6	V7	V8
		タグ貼付	梱包/積込	出荷	入荷	タグ貼付	梱包/積込	出荷
	eventType	ObjectEvent	Aggregation Event	ObjectEvent	ObjectEvent	ObjectEvent	Aggregation Event	ObjectEvent
	action	ADD	ADD	OBSERVE	OBSERVE	ADD	ADD	OBSERVE
What	parentID		grai1				grai1	
	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	grai1	grai1	sgtin4 sgtin5 sgtin6	sgtin4 sgtin5 sgtin6	grai1
Why	bizStep	commissioning	packing/ loading	shipping	receiving	commissioning	packing/ loading	shipping
	disposition	active	in_progress	in_transit	in_progress	active	in_progress	in_transit
	source			出荷元の SGLN	出荷元の SGLN			出荷元の SGLN
	destination			出荷先の SGLN	出荷先の SGLN			出荷先の SGLN

したがって、再利用資産を利用しながら EPCIS データ共有を行うためには、再利用資産の再投入を識別するための情報が必要である。次の表に再利用資産の再投入を識別するためのイベントを投入した例を示す。
この例では資産の返却(assets_returning/in_transit)および受領(assets_receiving/assets_returned)をイベントとして登録している。これらのイベントに指定している bizStep、disposition は CBV[v1.2.2]にて定義されていないので、今後日本のビジネスで必要になるか議論が必要である。

表 3-18 再利用資産の返却・受領イベントの追加

		出荷元事業者			出荷先事業者			出荷元事業者			
		V1	V2	V3	V4	V5	**V5.1**	**V5.2**	V6	V7	V8
		タグ貼付	梱包/積込	出荷	入荷	開梱/積降	**資産返却**	**資産受領**	タグ貼付	梱包/積込	出荷
	eventType	ObjectEvent	Aggregation Event	Object Event	Object Event	Aggregation Event	**Object Event**	**Object Event**	ObjectEvent	Aggregation Event	Object Event
	action	ADD	ADD	OBSERVE	OBSERVE	DELETE	**OBSERVE**	**OBSERVE**	ADD	ADD	OBSERVE
What	parentID		grai1			grai1				grai1	
	epcList	sgtin1 sgtin2 sgtin3	sgtin1 sgtin2 sgtin3	grai1	grai1	sgtin1 sgtin2 sgtin3	**grai1**	**grai1**	sgtin4 sgtin5 sgtin6	sgtin4 sgtin5 sgtin6	grai1
Why	bizStep	commissioning	packing/ loading	shipping	receiving	unpacking/ unloading	**assets_ returning**	**assets_ receiving**	commissioning	packing/ loading	shipping
	disposition	active	in_progress	in_transit	in_progress		**in_transit**	**assets_ returned**	active	in_progress	in_transit
	source			出荷元の SGLN	出荷元の SGLN		**出荷先の SGLN**	**出荷先の SGLN**			出荷元の SGLN
	destination			出荷先の SGLN	出荷先の SGLN		**出荷元の SGLN**	**出荷元の SGLN**			出荷先の SGLN

このようにイベントを追加したことで、V8 時点で関連イベントを検索する際、次のようにすることで、2 巡目以降も関連イベントを正しく判別することができるようになる。

1. grai1 で検索し、V2, V3, V4, V5, V5.1, V5.2, V7 を取得
   (V2, V3, V4, V5, V5.1, V5.2, V7)
2. V5.1 または V5.2 を取得したことによりそれ以前の grai1 を無効と判断し、イベント除外
   (V7)
3. V7 が AggregationEvent であるため、sgtin4,sgtin5,sgtin6 で検索し、V6 を取得
   (V6, V7)
4. 得られた関連イベントと出荷イベントが出荷先へ共有するイベントと判断
   (V6, V7, V8)

また、3 巡目以降は資産の返却・受領イベントが複数取得される可能性があるが、その場合は取得されたイベントの内、一番新しいイベント以前のイベントを無効とすればよい。

3.3.3. 中間事業者で複数事業者の物品が混載される場合の下流から上流へのデータ共有

卸事業者が複数の製造事業者から入荷した物品を再梱包により混載してから小売事業者へ出荷するような場合、再梱包を行った卸事業者は次の図に示すように、上流側への EPCIS データ共有を行う場合に各製造事業者が取り扱っている物品についての情報だけを共有することが望ましい。

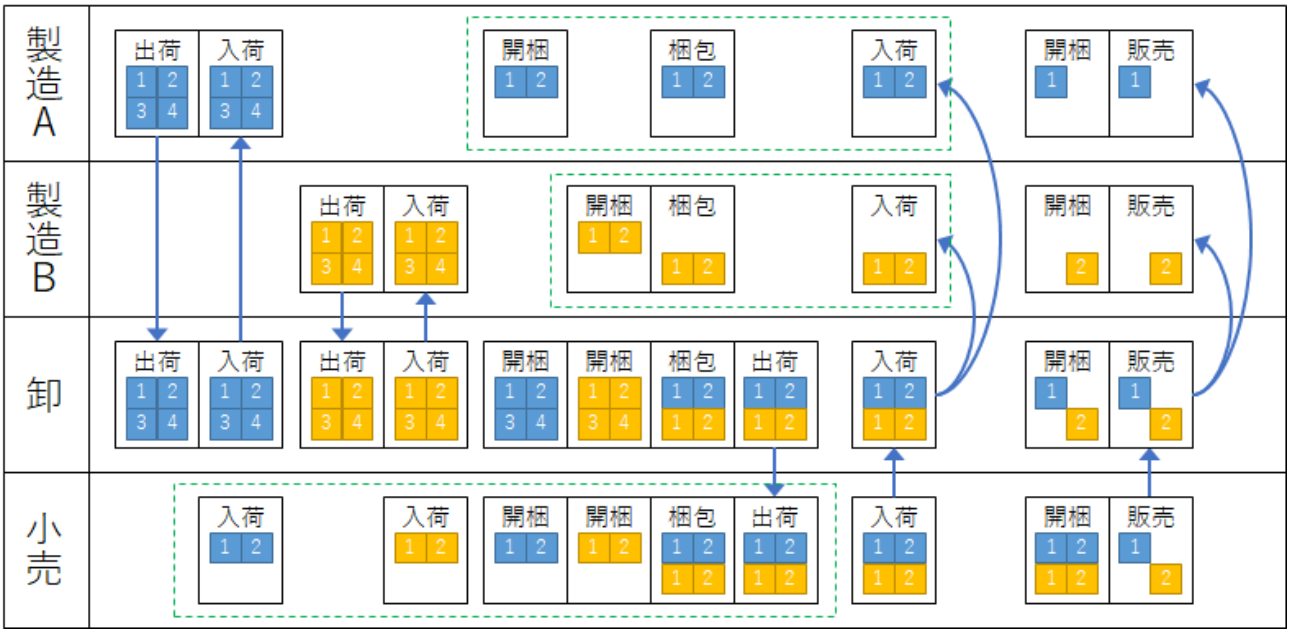

図 3-2 複数の製造事業者から入荷した物品を卸事業者が混載して小売事業者へ出荷する例

3.4. マスタデータの共有

製造事業者が保有している商品情報を小売事業者も利用したい場合など、イベントとともにマスタデータを共有したい場合がある。この場合、EPCIS リポジトリからマスタデータの検索を行いイベントとともにデータ共有を行ってもよい。マスタデータの共有を行う場合は、EPCISDocument および EPCISMasterDocument の二つに分けてデータ連携先 EPCIS のキャプチャ・インタフェースへデータを送信する。データ共有先すべてが EPCIS 標準仕様 V1.2 において示されている、EPCISDocument のヘッダにマスタデータを含めてもよいとされている仕様に対応しているならば一つの EPCISDocument にイベント及びマスタデータを格納して送信してもよい。次の表に商品マスタデータの例を示す。

表 3-19 商品マスタデータの例

type(Vocabulary) : urn:epcglobal:epcis:vtype:EPCClass		
id(VocabularyElement)	**id(attribute)**	**value(attribute)**
urn:epc:idpat:sgtin:0000001.000001.*	urn:epcglobal:cbv:mda: manufacturerOfTradeItemPartyName	メーカーX
	urn:epcglobal:cbv:mda:labelDescription	お茶 500ml
	urn:epcglobal:cbv:mda: regulatedProductName	緑茶(清涼飲料水)
urn:epc:idpat:sgtin:0000001.000002.*	urn:epcglobal:cbv:mda: manufacturerOfTradeItemPartyName	メーカーX
	urn:epcglobal:cbv:mda:labelDescription	水 500ml
	urn:epcglobal:cbv:mda: regulatedProductName	ミネラルウォーター

3.5. データ共有先の特定

EPCIS 間でデータ共有をする場合、通知されたイベントをもとに、どの事業者へデータを共有すればよいかを判断する必要がある。イベントサブスクライバからコールバックされる、データ共有の起点となるイベントについて、送信先を特定するための情報として source/destination の情報を含めるべきである。EPCIS データ連携ミドルウェアは、通知されたイベント情報の destination の情報から共有先となる事業者を特定し、送信先を判別する。
例えば、事業者およびその送信先情報は、次の表に示すように EPCIS リポジトリのマスタ情報のボキャブラリ型である SourceDestID(vtype="urn:epcglobal:epcis:vtype:SourceDest")についての情報として格納する。

表 3-20 データ共有先情報の例

type(Vocabulary) : urn:epcglobal:epcis:vtype:SourceDest		
id(VocabularyElement)	**id(attribute)**	**value(attribute)**
urn:epc:id:sgln:0000001.00001.0	http://www.example.co.jp/epcis/captureurl	http://www.manufacture.co.jp/epcis/capture
urn:epc:id:sgln:0000002.00001.0	http://www.example.co.jp/epcis/captureurl	http://www.distributor.co.jp/epcis/capture

このようなデータを各事業者が EPCIS のマスタデータとして事前に設定しておくことで、EPCIS データ連携ミドルウェアはイベントの destination に設定された相手先の SGLN をもとにデータ共有先のアドレスを決定できる。
attribute 要素の id 属性については任意の値を設定してよいが、EPCIS データ連携ミドルウェアがデータ共有先アドレスの識別子であることを認識できる値にする必要がある。

3.5.1. データ共有時のセキュリティ

データ共有のために公開されている EPCIS キャプチャ・インタフェースがセキュリティで保護されている場合、EPCIS データ連携ミドルウェアは共有先別に適切なセキュリティで保護された接続を行い、データを送信する必要がある。そのため、接続先情報には、表 3-20 で示した送信先アドレスの情報のほかにセキュリティ方式を指定する識別子および指定されたセキュリティ方式に必要な情報を追加で格納してもよい。接続方法については事業者間であらかじめ確認しておく必要がある。attribute 要素の id 属性については送信先 URL の情報と同様に任意の値を設定してよいが、EPCIS データ連携ミドルウェアが認識できる値にする必要がある。次の表にセキュリティ情報を追加したマスタデータの例を示す。

表 3-21 接続のためのセキュリティ情報を付加したデータ共有先情報の例

type(Vocabulary) : urn:epcglobal:epcis:vtype:SourceDest		
id(VocabularyElement)	**id(attribute)**	**value(attribute)**
urn:epc:id:sgln:0000001.00001.0	http://www.example.co.jp/epcis/captureurl	http://www.manufacture.co.jp/epcis/capture
	http://www.example.co.jp/epcis/authtype	Basic(*)
	http://www.example.co.jp/epcis/capture_user	user1(*)
	http://www.example.co.jp/epcis/capture_password	pass1(*)
urn:epc:id:sgln:0000002.00001.0	http://www.example.co.jp/epcis/captureurl	http://www.distributor.co.jp/epcis/capture
	http://www.example.co.jp/epcis/authtype	Basic(*)
	http://www.example.co.jp/epcis/capture_user	user2(*)
	http://www.example.co.jp/epcis/capture_password	pass2(*)

(*)ここでは説明のため平文でセキュリティ情報を記載しているが、運用上は暗号化されていることが望ましい。

3.5.2. EPCIS 未導入の事業者の考慮

EPCIS はすべての事業者が導入しているとは限らず、サプライチェーンの経路上に EPCIS 未導入の事業者が存在する場合、その事業者との取引時点でデータ共有が途切れてしまう。
例えば、次の図に示すように製造、一次卸、二次卸、小売の四事業者がサプライチェーンの経路上にあり、二次卸が EPCIS を導入していない場合を考える。

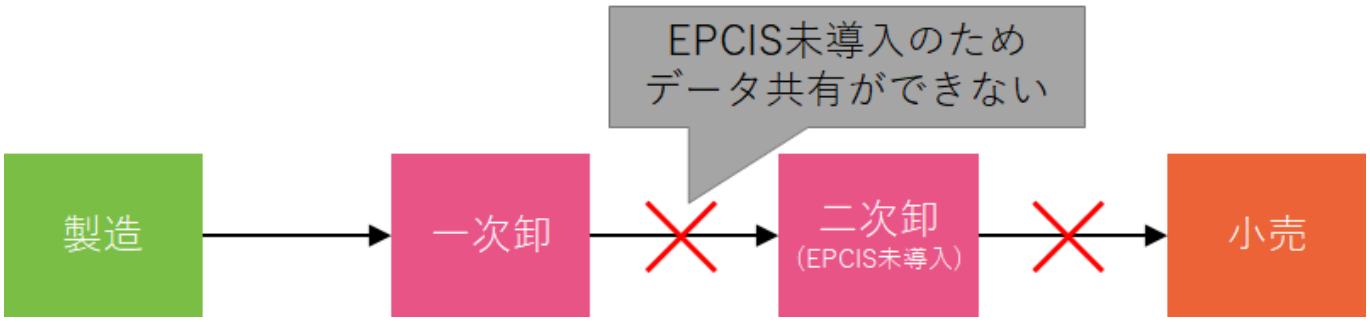

図 3-3 サプライチェーンの経路上に EPCIS 未導入の事業者が存在する場合

この時、EPCIS を導入している各事業者は共有先が存在しない場合にイベント情報から判別できる範囲で代替の共有先を特定しデータの共有をすることが望ましい。
図 3-3 のような場合、一次卸事業者から二次卸事業者へ出荷する時点では二次卸がどの小売事業者へ物品を販売するか不明なため、上流から下流へ EPCIS 未導入の事業者を飛ばしてデータ共有することは不可能である。このことからデータ共有の経路が分断されている場合、上流から下流への共有は不可能であることがわかる。

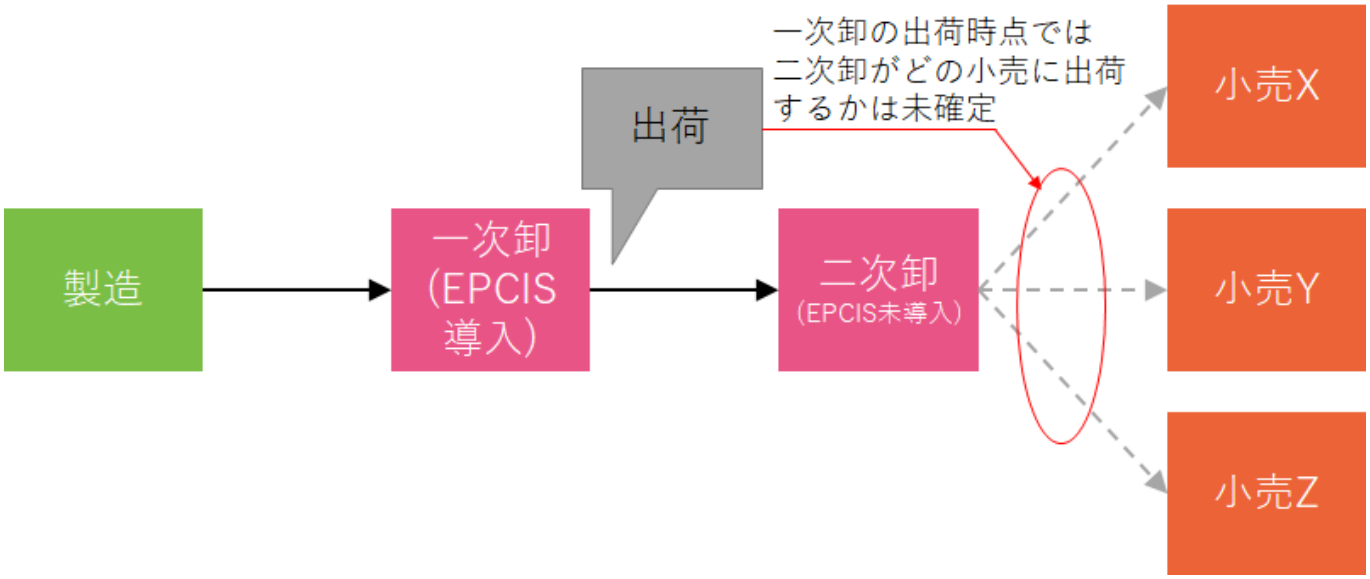

図 3-4 EPCIS 未導入の事業者が存在する場合の上流から下流への共有

しかしながら、小売事業者が二次卸事業者から物品を入荷した時、どのような経路で物品が流通されてきたかは判別できないが、実際に入荷した物品自体は識別が可能であるため、物品に付加された EPC より物品の製造事業者を特定できる可能性があり、下流から上流へはデータ共有できる場合がある。

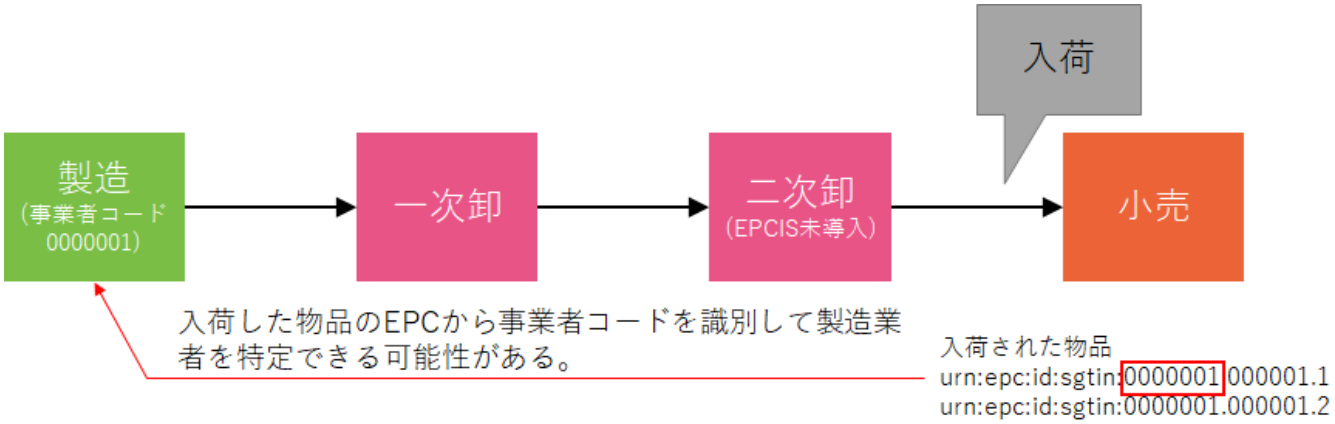

図 3-5 EPCIS 未導入の事業者が存在する場合の下流から上流への共有

このような場合、サプライチェーンの当事者間でデータ共有の合意があれば、卸事業者を経由せず、小売事業者から製造事業者へ入荷イベントや販売イベントなどのデータを直接共有してもよい。

4. 参考文献

1. EPCIS（EPC インフォメーション・サービス）一般財団法人流通システム開発センター
http://www.dsri.jp/standard/epc/epcis.html

2. EPC/RFID

https://www.gs1.org/standards/epc-rfid

3. EPCIS and Core Business Vocabulary (CBV)
https://www.gs1.org/standards/epcis

4. EPCIS 及び CBV 導入ガイドライン 日本語参考訳
http://www.dsri.jp/standard/epc/pdf/EPCIS-Guideline-1.2_J_170522.pdf

5. EPC Information Service(EPCIS) Standard Release 1.2
https://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf

6. Core Business Vocabulary Standard Release 1.2.2,
https://www.gs1.org/sites/default/files/docs/epc/CBV-Standard-1-2-2-r-2017-10-12.pdf

A. 付録：XML の例

A.1 表 3-1 のサブスクライブ要求の XML

```
<epcisq:Subscribe
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcisq="urn:epcglobal:epcis-query:xsd:1">
  <queryName>SimpleEventQuery</queryName>
  <params>
    <param>
      <name>eventType</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>ObjectEvent</string>
      </value>
    </param>
    <param>
      <name>EQ_bizStep</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>urn:epcglobal:cbv:bizstep:shipping</string>
      </value>
    </param>
  </params>
  <dest>http://www.example.co.jp/epcis/queryCallback</dest>
  <controls>
    <schedule>
      <second>0,30</second>
    </schedule>
    <reportIfEmpty>false</reportIfEmpty>
  </controls>
  <subscriptionID>shipping001</subscriptionID>
</epcisq:Subscribe>
```

A.2 表 3-2 のサブスクライブ要求の XML

```
<epcisq:Subscribe
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcisq="urn:epcglobal:epcis-query:xsd:1">
  <queryName>SimpleEventQuery</queryName>
  <params>
    <param>
      <name>eventType</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>ObjectEvent</string>
      </value>
    </param>
    <param>
      <name>EQ_bizStep</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>urn:epcglobal:cbv:bizstep:receiving</string>
      </value>
    </param>
  </params>
  <dest>http://www.example.co.jp/epcis/queryCallback</dest>
  <controls>
    <schedule>
      <second>0,30</second>
    </schedule>
    <reportIfEmpty>false</reportIfEmpty>
  </controls>
  <subscriptionID>receiving001</subscriptionID>
</epcisq:Subscribe>
```

A.3 表 3-3 のサブスクライブ要求の XML

```
<epcisq:Subscribe
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcisq="urn:epcglobal:epcis-query:xsd:1">
  <queryName>SimpleEventQuery</queryName>
  <params>
    <param>
      <name>eventType</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>ObjectEvent</string>
      </value>
    </param>
    <param>
      <name>EQ_bizStep</name>
      <value xsi:type="epcisq:ArrayOfString">
        <string>urn:epcglobal:cbv:bizstep:retail_selling</string>
      </value>
    </param>
  </params>
  <dest>http://www.example.co.jp/epcis/queryCallback</dest>
  <controls>
    <schedule>
      <second>0,30</second>
    </schedule>
    <reportIfEmpty>false</reportIfEmpty>
  </controls>
  <subscriptionID>retail_selling001</subscriptionID>
</epcisq:Subscribe>
```

A.4 表 3-19 のマスタデータの XML

```
<epcismd:EPCISMasterDataDocument
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcismd="urn:epcglobal:epcis-masterdata:xsd:1"
  schemaVersion="1.2" creationDate="2019-03-15T00:00:00 Z" >
  <EPCISBody>
    <VocabularyList>
      <Vocabulary type="urn:epcglobal:epcis:vtype:EPCClass">
        <VocabularyElementList>
          <VocabularyElement id="urn:epc:idpat:sgtin:0000001.000001.*">
            <attribute id="urn:epcglobal:cbv:mda:manufacturerOfTradeItemPartyName">メーカーX
</attribute>
            <attribute id="urn:epcglobal:cbv:mda:labelDescription">お茶　500ml</attribute>
            <attribute id="urn:epcglobal:cbv:mda:regulatedProductName">緑茶</attribute>
          </VocabularyElement>
          <VocabularyElement id="urn:epc:idpat:sgtin:0000001.000002.*">
            <attribute id="urn:epcglobal:cbv:mda:manufacturerOfTradeItemPartyName">メーカーX
</attribute>
            <attribute id="urn:epcglobal:cbv:mda:labelDescription">水　500ml</attribute>
            <attribute id="urn:epcglobal:cbv:mda:regulatedProductName">ミネラルウォーター</attribute>
          </VocabularyElement>
        </VocabularyElementList>
      </Vocabulary>
    </VocabularyList>
  </EPCISBody>
</epcismd:EPCISMasterDataDocument>
```

A.5 表 3-20 のマスタデータの XML

```
<epcismd:EPCISMasterDataDocument
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcismd="urn:epcglobal:epcis-masterdata:xsd:1"
  schemaVersion="1.2" creationDate="2019-03-15T00:00:00Z">
  <EPCISBody>
    <VocabularyList>
      <Vocabulary type="urn:epcglobal:epcis:vtype:SourceDest">
        <VocabularyElementList>
          <VocabularyElement id="urn:epc:id:sgtin:0000001.00001.0">
            <attribute
id="http://www.example.co.jp/epcis/captureurl">http://www.manufacture.co.jp/epcis/capture</attribute>
          </VocabularyElement>
        </VocabularyElementList>
        <VocabularyElementList>
          <VocabularyElement id="urn:epc:id:sgtin:0000002.00001.0">
            <attribute
id="http://www.example.co.jp/epcis/captureurl">http://www.distributor.co.jp/epcis/capture</attribute>
          </VocabularyElement>
        </VocabularyElementList>
      </Vocabulary>
    </VocabularyList>
  </EPCISBody>
</epcismd:EPCISMasterDataDocument>
```

A.6 表 3-21 のマスタデータの XML

```
<epcismd:EPCISMasterDataDocument
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:epcismd="urn:epcglobal:epcis-masterdata:xsd:1"
  schemaVersion="1.2" creationDate="2019-03-15T00:00:00Z">
  <EPCISBody>
    <VocabularyList>
      <Vocabulary type="urn:epcglobal:epcis:vtype:SourceDest">
        <VocabularyElementList>
          <VocabularyElement id="urn:epc:id:sgtin:0000001.00001.0">
            <attribute
id="http://www.example.co.jp/epcis/captureurl">http://www.manufacture.co.jp/epcis/capture</attribute>
            <attribute id="http://www.example.co.jp/epcis/authtype">Basic</attribute>
            <attribute id="http://www.example.co.jp/epcis/captur_user">user1</attribute>
            <attribute id="http://www.example.co.jp/epcis/captur_password">pass1</attribute>
          </VocabularyElement>
        </VocabularyElementList>
        <VocabularyElementList>
          <VocabularyElement id="urn:epc:id:sgtin:0000002.00001.0">
            <attribute
id="http://www.example.co.jp/epcis/captureurl">http://www.distributor.co.jp/epcis/capture</attribute>
            <attribute id="http://www.example.co.jp/epcis/authtype">Basic</attribute>
            <attribute id="http://www.example.co.jp/epcis/captur_user">user2</attribute>
            <attribute id="http://www.example.co.jp/epcis/captur_password">pass2</attribute>
          </VocabularyElement>
        </VocabularyElementList>
      </Vocabulary>
    </VocabularyList>
  </EPCISBody>
</epcismd:EPCISMasterDataDocument>
```

B. 付録：クエリコールバックサンプル実装

B.1 出荷イベントを共有する例

次に示すコードは C# 7.0 による実装例である。
GS1 にて公開されているデータスキーマ(xsd ファイル)を用いて MS XML Schema Definition Tool(xsd.exe)によりクラスを自動生成していることを前提にしている。本サンプルでは出荷イベントを上流から下流の方向へ情報を共有する。出荷時に送信先の EPCIS は出荷物に関する一切の情報を持っていないため、送信元 EPCIS にある出荷イベントに関連するすべてのイベントを共有する。
なお、サンプルは処理全体を把握しやすくするため、エラー処理、実装の省略・簡略化等を行っている。

```
// インタフェース定義
public interface CoreCaptureService
{
    void capture(List<EPCISEventType> @event);
    // <<extension point>>
    // マスタデータのキャプチャを行う場合はキャプチャ・インタフェースを拡張する必要がある
    void capture(List<VocabularyType> vocabularies);
}

public interface EPCISQueryControlInterface
{
    void subscribe(
        string queryName, QueryParam[] @params, Uri dest,
        SubscriptionControls controls, string subscriptionID);
    void unsubscribe(string subscriptionID);
    QueryResults poll(string queryName, QueryParam[] @params);
    ArrayOfString getQueryNames();
    ArrayOfString getSubscriptionIDs(string queryName);
    string getStandardVersion();
    string getVendorVersion();
}

public interface EPCISQueryCallbackInterface
{
    void callbackResults(QueryResults resultData);
    void callbackQueryTooLargeException(QueryTooLargeException e);
    void callbackImplementationException(ImplementationException e);
}
```

```
// EPCISクエリコールバック・インタフェース実装
public class EPCISQueryCallbackInterfaceImplementation : EPCISQueryCallbackInterface
{
    public void callbackResults(QueryResults resultData)
    {
        // NOTE: 固定の実装でsubscriptionIDにより処理を分岐しているが
        //       外部設定(ファイルやデータベースなど)により動的に実装してもよい。
        switch (resultData.subscriptionID)
        {
            case "shipping":
                var shippingProcess = new ProcessForShippingEvents();
                shippingProcess.Execute(resultData);
                break;
            // NOTE: ここにその他のサブスクリプションIDに対応する実装を記述してもよい。
            // case "receiving":
            //     var receivingProcess = new ProcessForReceivingEvents();
            //     receivingProcess.Execute(resultData);
            //     break;
            // case "retail_selling":
            //     var retailSellingProcess = new ProcessForRetailSellingEvents();
            //     retailSellingProcess.Execute(resultData);
            //     break;
        }
    }

    public void callbackQueryTooLargeException(QueryTooLargeException e)
    {
        // TODO: ここにQueryTooLargeExceptionがコールバックされた時の処理を実装する。
    }

    public void callbackImplementationException(ImplementationException e)
    {
        // TODO: ここにImplementationExceptionがコールバックされた時の処理を実装する。
    }
}
```

```
// サブスクリプションID別実装(出荷)
public class ProcessForShippingEvents
{
    // TODO: データ共有のために接続するEPCISクエリコントロール・インタフェースのURL
    //       実際には実行環境に合わせて固定値、外部設定等から決定する必要がある。
    private string queryControlInterfaceUrl = "http://www.example.co.jp/epcis/querycontrol";
    private EPCISQueryControlInterface epcisQueryControlInterface;

    public ProcessForShippingEvents()
    {
        epcisQueryControlInterface = CreateQueryControlInterfaceInstance(queryControlInterfaceUrl);
    }

    public void Execute(QueryResults resultData)
    {
        var eventList = (EventListType)resultData.resultsBody.Item;

        // トリガイベントごとに関連イベントを収集し、キャプチャ・インタフェースへイベントを送信する。
        foreach (object item in eventList.Items)
        {
            var relativeEvents = new List<EPCISEventType>();

            var @event = item as EPCISEventType;

            // EPCISのマスタデータから宛先となるキャプチャ・インタフェースのURLを検索する
            string owningPartyValue = GetDestinationOwningPartyValue(@event);
            string destCaptureUrl = SelectDestinationCaptureInterfaceUrl(owningPartyValue);

            // 送信先が見つからない場合は処理終了
            if (string.IsNullOrWhiteSpace(destCaptureUrl)) { continue; }

            // 関連イベントを収集する
            relativeEvents.Add(@event);
            relativeEvents.AddRange(SelectRelativeEvents(@event));

            // 重複がある場合は排除する
            relativeEvents = DistinctEvents(relativeEvents);

            // eventTimeの昇順でソートする
            relativeEvents.Sort((e1, e2) => e1.eventTime.CompareTo(e2.eventTime));

            // 必要な場合、イベントのフィルタリングを行う
            relativeEvents = FilterEvents(relativeEvents);

            // 必要な場合、マスタデータの検索を行う
            List<VocabularyType> vocabularies = SelectMasterData(relativeEvents);

            // イベントを送信する
            CoreCaptureService coreCaptureService = CreateCoreCaptureServiceInstance(destCaptureUrl);
            coreCaptureService.capture(relativeEvents);

            // マスタデータが存在しなければ処理終了
            if (vocabularies?.Any() == false) { continue; }

            // マスタデータを送信する
            coreCaptureService.capture(vocabularies);
        }
    }
```

```
private CoreCaptureService CreateCoreCaptureServiceInstance(string url)
{
    // TODO: ここで指定されたURLを送信先とするキャプチャ・インタフェースへ接続するための
    //       クライアントクラスのインスタンスを生成する。
    throw new NotImplementedException();
}

private EPCISQueryControlInterface CreateQueryControlInterfaceInstance(string url)
{
    // TODO: ここで指定されたURLを送信先とするクエリコントロール・インタフェースへ接続するための
    //       クライアントクラスのインスタンスを生成する。
    throw new NotImplementedException();
}

// EPCISEventType型のオブジェクトからdestinationを取得する。
private string GetDestinationOwningPartyValue(EPCISEventType @event)
{
    SourceDestType[] destinationList = GetDestinationList(@event);
    SourceDestType destOwningParty = (from d in destinationList
                                      where d.type == "urn:epcglobal:cbv:sdt:owning_party"
                                      select d).SingleOrDefault();
    return destOwningParty?.Value;
}

// EPCISEentType型のオブジェクトからdestinationListを取得する。
private SourceDestType[] GetDestinationList(EPCISEventType @event)
{
    switch (@event)
    {
        case ObjectEventType objectEvent:
            return objectEvent.destinationList;
        case AggregationEventType aggregationEvent:
            return aggregationEvent.destinationList;
        default:
            return null;
    }
}

// 取引先のSGLNを指定してEPCISのマスタデータからキャプチャ・インタフェースのURLを検索する。
// 本サンプルでは属性のidが"http://www.example.co.jp/epcis/captururl"であるものとしている。
private string SelectDestinationCaptureInterfaceUrl(string owningPartyValue)
{
    if (string.IsNullOrEmpty(owningPartyValue)) { return null; }

    var queryName = "SimpleMasterDataQuery";
    var @params = new QueryParam[]
    {
        new QueryParam()
        {
            name = "vocabularyName",
            value = new ArrayOfString()
            {
                @string = new string[] { "urn:epcglobal:epcis:vtype:SourceDest" }
            }
        },
        new QueryParam() { name = "includeAttributes", value = true },
```

```
            new QueryParam() { name = "includeChildren", value = false },
            new QueryParam()
            {
                name = "EQ_name",
                value = new ArrayOfString() { @string = new string[] { owningPartyValue } }
            },
        };
        QueryResults queryResults = epcisQueryControlInterface.poll(queryName, @params);
        var vocabularyList = queryResults.resultsBody.Item as VocabularyListType;
        var destUrl = (from a in vocabularyList.Vocabulary[0].VocabularyElementList[0].attribute
                       where a.id == "http://www.example.co.jp/epcis/captureurl"
                       select a.Text[0]).SingleOrDefault();
        return destUrl;
    }

    // EPCISのイベントデータから指定されたイベントに関連するイベントを検索する。
    private List<EPCISEventType> SelectRelativeEvents(EPCISEventType @event)
    {
        var relativeEvents = new List<EPCISEventType>();

        var queryName = "SimpleEventQuery";
        var @params = new QueryParam[]
        {
            new QueryParam()
            {
                name = "eventType",
                value = new ArrayOfString()
                {
                    @string = new string[] { "ObjectEvent", "AggregationEvent" }
                }
            },
            new QueryParam() { name = "LT_eventTime", value =  @event.eventTime },
            new QueryParam()
            {
                name = "MATCH_anyEPC",
                value = new ArrayOfString() { @string = GetEPCList(@event).Select(epc=>epc.Value).ToArray() }
            },
        };
        QueryResults queryResults = epcisQueryControlInterface.poll(queryName, @params);
        var relativeEventList = (EventListType)queryResults.resultsBody.Item;

        relativeEvents.AddRange(relativeEventList.Items.Select(item => (EPCISEventType)item));

        // 検索結果にAggregationEvnetが含まれている場合は
        // そのAggregationEventを起点とした関連イベントを再帰的に検索する
        var aggregationEvents = from re in relativeEventList.Items
                                where re is AggregationEventType
                                select (AggregationEventType)re;
        foreach (var aggregationEvent in aggregationEvents)
        {
            relativeEvents.AddRange(SelectRelativeEvents(aggregationEvent));
        }

        return relativeEvents;
    }
```

```
// EPCISEventType型のオブジェクトに含まれる関連イベントの検索条件となるEPCを取得する
private EPC[] GetEPCList(EPCISEventType @event)
{
    switch (@event)
    {
        case ObjectEventType objectEvent:
            return objectEvent.epcList;
        case AggregationEventType aggregationEvent:
            return
                aggregationEvent.childEPCs.
                Concat(new EPC[] { new EPC() { Value = aggregationEvent.parentID } }).
                ToArray();
        default:
            return null;
    }
}

// イベントの重複を排除する。
private List<EPCISEventType> DistinctEvents(List<EPCISEventType> events)
{
    // TODO: イベント重複排除処理を実装する。
    return events;
}

// イベントのフィルタリングを行う。
private List<EPCISEventType> FilterEvents(List<EPCISEventType> events)
{
    // TODO: イベントフィルタ処理を実装する。
    return events;
}

// マスタデータの検索を行う。
private List<VocabularyType> SelectMasterData(List<EPCISEventType> events)
{
    // TODO: マスタデータ検索処理を実装する。
    // 例：商品マスタを検索する
    var sgtinList = (from ev in events select GetEPCList(ev)).
        SelectMany(epcArray => epcArray).
        Where(epc => epc.Value.StartsWith("urn:epc:id:sgtin:")).
        Select(epc => epc.Value).
        Distinct();

    var queryName = "SimpleMasterDataQuery";
    var @params = new QueryParam[]
    {
        new QueryParam()
        {
            name = "vocabularyName",
            value = new ArrayOfString()
            {
                @string = new string[] { "urn:epcglobal:epcis:vtype:EPCClass" }
            }
        },
        new QueryParam() { name = "includeAttributes", value = true },
        new QueryParam() { name = "includeChildren", value = false },
```

```
            new QueryParam()
            {
                name = "EQ_name",
                value = new ArrayOfString()
                {
                    @string = ConvertFromIndividualToClassSGTIN(sgtinList.ToArray())
                }
            },
            // マスタデータの内、一部だけを取得する場合は次の条件を有効にする。
            // new QueryParam()
            // {
            //     name = "attributeNames",
            //     value = new ArrayOfString() { @string = new string[] { /* TODO: set field names */ } }
            // },
        };
        QueryResults queryResults = epcisQueryControlInterface.poll(queryName, @params);
        var vocabularyList = queryResults?.resultsBody?.Item as VocabularyListType;
        return vocabularyList?.Vocabulary?.ToList();
    }

    private string[] ConvertFromIndividualToClassSGTIN (string[] individualSGTINs)
    {
        var classSGTINs = new List<string>();
        foreach (string individualSGTIN in individualSGTINs)
        {
            // NOTE: urn:epc:id:sgtin:xxxxxxx.xxxxxx.xxxxxxxx -> urn:epc:idpat:sgtin:xxxxxxx.xxxxxx.*
            string classSGTIN = "urn:epc:idpat:sgtin:";
            classSGTIN += individualSGTIN.Substring(
                individualSGTIN.LastIndexOf(':') + 1,
                individualSGTIN.LastIndexOf('.') - individualSGTIN.LastIndexOf(':'));
            classSGTIN += "*";
            classSGTINs.Add(classSGTIN);
        }
        return classSGTINs.Distinct().ToArray();
    }
}
```

変更履歴

日付	変更箇所	内容