ASPIRE Programmable Language and Engine

Athens Information Technology

Agenda

- ASPIRE Programmable Language (APDL)
- ASPIRE Programmable Engine (APE)

ASPIRE Programmable Language

ASPIRE Programmable Language (1)

- Goal
  - To create a programmability language for RFID solutions
  - Possible to be standardized
  - XML based
  - Process Based (e.g. XPDL)

ASPIRE Programmable Language (2)

- Why a programmable language?
  - Easier to use than General Purpose Languages (GPL's: C, Java, etc.)
  - Intends to increase productivity
    - Users/target developers do not need to learn any GPL
  - Increased expressiveness of domain-specific notations

ASPIRE Programmable Language (3)

- Why not a GPL?
  - Much harder to achieve the expressiveness of domain-specific notations
  - GPL source code tends to be too complex

ASPIRE Programmable Language (4)

- A combination of
  - Logical Readers Specs
  - ECSpecs
  - Master Data Document
  - Middleware Management/Configuration Data (BEG, Connector)
  - Business Workflow data
    - With design data for visualization

Business Process Management

- Base Idea: A product or service is the outcome of a number of activities
- Organize and execute these activities

Business Process Modeling Notation

- BPMN is a standard modeling notation
  - Eases understanding of business procedures
  - Enables communication of these procedures in a standardized manner
  - Provides graphical notation for Business Process Diagrams (BPD)
  - Provides bindings to block-structured process execution languages (e.g. BPML, BPEL-WS)

Programming languages for BPM (1)

- BPML
  - Meta-language for modeling business processes
  - Supports entities
  - Abstract and executable processes
  - XML grammar

Programming languages for BPM (2)

- BPEL
  - Orchestration, not choreography
  - Uses WSDL
  - Includes structured-programming constructs
    - If-then-else, while, flow (for parallel execution)
  - Supports a scoping system
  - Serialized scopes to control concurrent access to variables

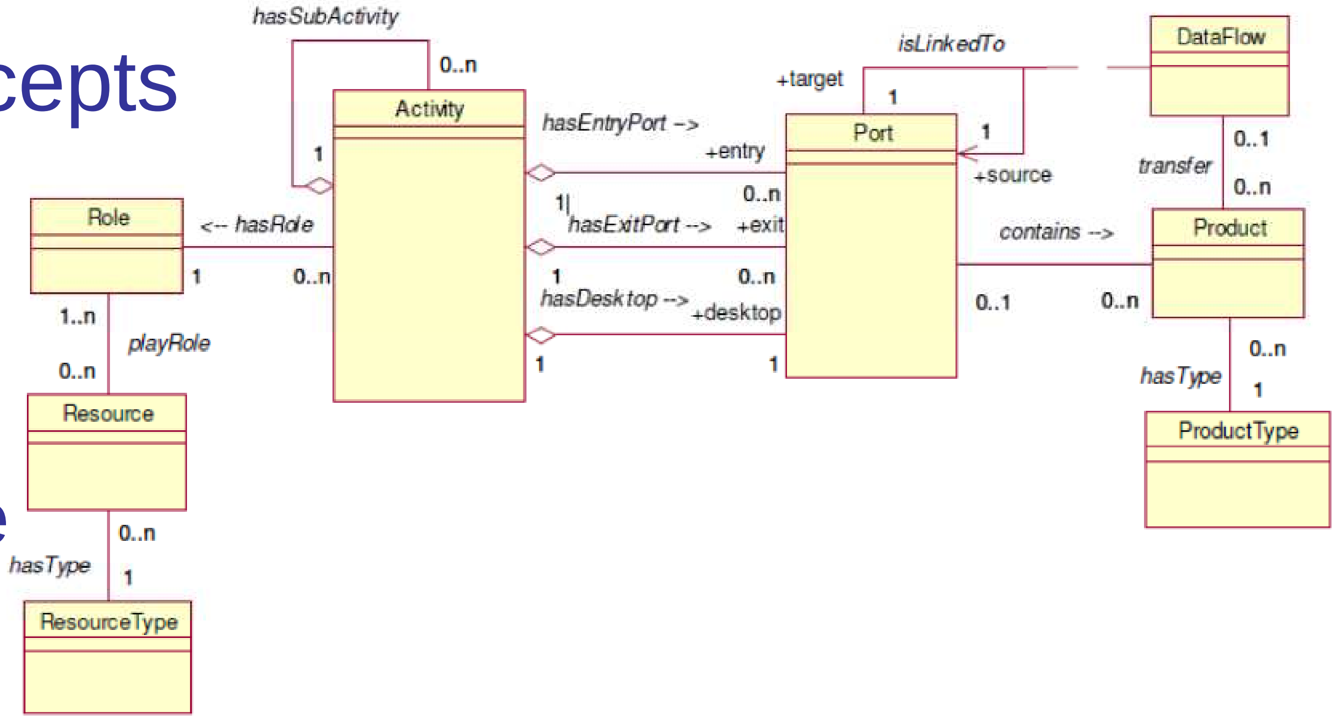
Programming languages for BPM (3)

- YAWL
  - Open source (LGPL)
  - Highly expressive language
  - Bases on Petri nets formalisms (although not completely compatible)

Programming languages for BPM (4)

- APEL
- Basic concepts
  - Activity
  - Product
  - Port
  - Resource
  - Dataflow

APEL Metamodel



Programming languages for BPM (5)

- XPDL
  - The serialization format for BPMN
  - XML-based
  - Includes graphical descriptions
  - Standard supported by WfMC

Programming languages for BPM (6)

- Decision: XPDL
  - BPEL does not define a graphical diagram
  - Only one implementation for YAWL and APEL, respectively
- AspireRfid Process Description Language (APDL) is based on XPDL v1.0

A Closer View on XPDL (1)

- XPDL main elements
  - Package
    - Container holding the other elements
  - Application
    - used to specify the applications/tools invoked by the workflow processes
  - Workflow-Process
    - Defines workflow processes or parts of workflow processes. Composed by

A Closer View on XPDL (2)

- XPDL main elements
  - Workflow-Process is composed by
    - Activity: the basic building block of a workflow process definition
    - Transition
      - connects elements of type Activity
      - Three types: Route, Implementation, and BlockActivity
  - Participant
    - Specifies participants in the workflow
    - 6 types: ResourceSet, Resource, Role, OrganizationalUnit, Human, and System

A Closer View on XPDL (3)

- XPDL main elements
  - DataField and DataType
    - specify workflow relevant data
    - Data is used to make decisions or to refer to data outside of the workflow, and is passed between activities and subflows

APDL Description (1)

- Open Loop Composite Business Process (<OLCBProc/>) consists of
  – Closed Loop Composite Business Process (CLCBProc/>) describes
    - The supply chain scenario
    - The Transitions

APDL Description (2)

- <CLCBProc/>) consists of
  - Elementary Business Process (<EBProc/>)
    - Basic configuration variables
    - Workflow graphical representation variables (x/y coordinates)
    - Datafields (<DataField/>) that include
      - The transactions required ECSpec
      - The transactions required LRSpec
      - And the transactions required Master Data
  - Transitions objects (<Transitions/>)

Programmable Meta-Language Structure (1)

- The OLCBProc Element

```
<xs:complexType name="OLCBProc">
    <xs:sequence>
        <xs:element ref="apdl:CLCBProc" />
        <xs:element ref="apdl:Transitions" />
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:anyURI" />
    <xs:attribute name="name" use="required" type="xs:NCName" />
</xs:complexType>
```

- Description

Name	Description
CLCBProc	Close Loop Composite Business Process (see here)
Transitions	CLCBProc Transitions (see here)
id	The CLCBProc’s ID
name	The CLCBProc’s Name

Programmable Meta-Language Structure (2)

- The CLCBProc Element

```
<xs:complexType name="CLCBProc">
    <xs:sequence>
        <xs:element ref="apdl:Description" />
        <xs:element maxOccurs="unbounded" ref="apdl:EBProc" />
        <xs:element ref="apdl:Transitions" />
    </xs:sequence>
    <xs:attribute name="id" use="required" type="xs:anyURI" />
    <xs:attribute name="name" use="required" type="xs:NCName" />
</xs:complexType>
```

- Description

Name	Description
Description	The description of the CLCBProc
EBProc	Elementary Business Process (see here)
Transitions	The Transitions description (see here)
id	The CLCBProc's ID
name	The CLCBProc's Name

Programmable Meta-Language Structure (3)

- The EBProc Element

```
<xs:complexType name="EBProc">
    <xs:sequence>
        <xs:element ref="apdl:Description" />
        <xs:element ref="apdl:TransitionRestrictions" />
        <xs:element ref="apdl:ExtendedAttributes" />
        <xs:element minOccurs="1" maxOccurs="4" ref="apdl:DataFields" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:anyURI" />
    <xs:attribute name="name" type="xs:NCName" />
</xs:complexType>
```

- Description

Name	Description
Description	The EBProc's Description
TransitionRestrictions	The EBProc's Transition Restrictions (see here)
ExtendedAttributes	The EBProc's Extended Attributes (see here)
DataFields	The EBProc's Data Fields (see here)
Id	The EBProc's ID
name	The EBProc's Name

Programmable Meta-Language Structure (4)

- The TransitionRestrictions Element

```
<xs:element name="TransitionRestrictions">
    <xs:complexType>
        <xs:sequence>
    <xs:element ref="xpdl:TransitionRestriction"
                          minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- Description

Name	Description
TransitionRestriction	XPDL TransitionRestriction (XPDL V1.0 [4])

Programmable Meta-Language Structure (5)

- The ExtendedAttributes Element

```
<xs:element name="ExtendedAttributes">
    <xs:complexType>
        <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="apdl:ExtendedAttribute" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- Description

Name	Description
ExtendedAttribute	The EBProc Extended Attribute (see here)

Programmable Meta-Language Structure (6)

- The ExtendedAttribute Element

```
<xs:element name="ExtendedAttribute">
    <xs:complexType>
        <xs:attribute name="name" use="required" type="xs:NCName" />
        <xs:attribute name="value" use="required" type="xs:string" />
    </xs:complexType>
</xs:element>
```

- Description

Name	Description
name	The Extended Attribute’s name
value	The Extended Attribute’s value

Programmable Meta-Language Structure (7)

- The DataFields Element

```
<xs:element name="DataFields">
    <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" ref="apdl:DataField" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- Description

Name	Description
DataField	The EBProc Data list (see here)

Programmable Meta-Language Structure (8)

- The DataField Element

```
<xs:element name="DataField">
    <xs:complexType>
        <xs:choice>
            <xs:element ref="apdl:ECSpec" />
            <xs:element ref="apdl:EPCISMasterDataDocument" />
            <xs:element ref="apdl:LRSPec" />
        </xs:choice>
        <xs:attribute name="id" use="optional" type="xs:anyURI" />
        <xs:attribute name="name" use="optional" type="xs:NCName" />
        <xs:attribute name="type" use="optional" type="xs:NCName" />
    </xs:complexType>
</xs:element>
```

- Description

Name	Description
ECSpec	EBProc’s ECSpec (see here)
EPCISMasterDataDocument	EBProc’s EPCIS Master Data Document (see here)
LRSPec	EBProc’s LRSpec (see here)
id	DataField ID
name	DataField Name
type	DataField Type

Programmable Meta-Language Structure (9)

- The EPCISMasterDataDocument Element

```
<xs:element name="EPCISMasterDataDocument"
            type="epcismd:EPCISMasterDataDocumentType">
</xs:element>
```

- Description

Attribute Name	Attribute URI
EventName	urn:epcglobal:epcis:mda:event_name
EventType	urn:epcglobal:epcis:mda:event_type
BusinessStep	urn:epcglobal:epcis:mda:business_step
BusinessLocation	urn:epcglobal:epcis:mda:business_location
Disposition	urn:epcglobal:epcis:mda:disposition
ReadPoint	urn:epcglobal:epcis:mda:read_point
TransactionType	urn:epcglobal:epcis:mda:transaction_type
Action	urn:epcglobal:epcis:mda:action

Programmable Meta-Language Structure (10)

- The ECSpec Element

```
<xs:element name="ECSpec" type="ale:ECSpec"></xs:element>
```

Programmable Meta-Language Structure (11)

- The LRSpec Element

```
<xs:element name="LRSPec" type="alelr:LRSpec"></xs:element>
```

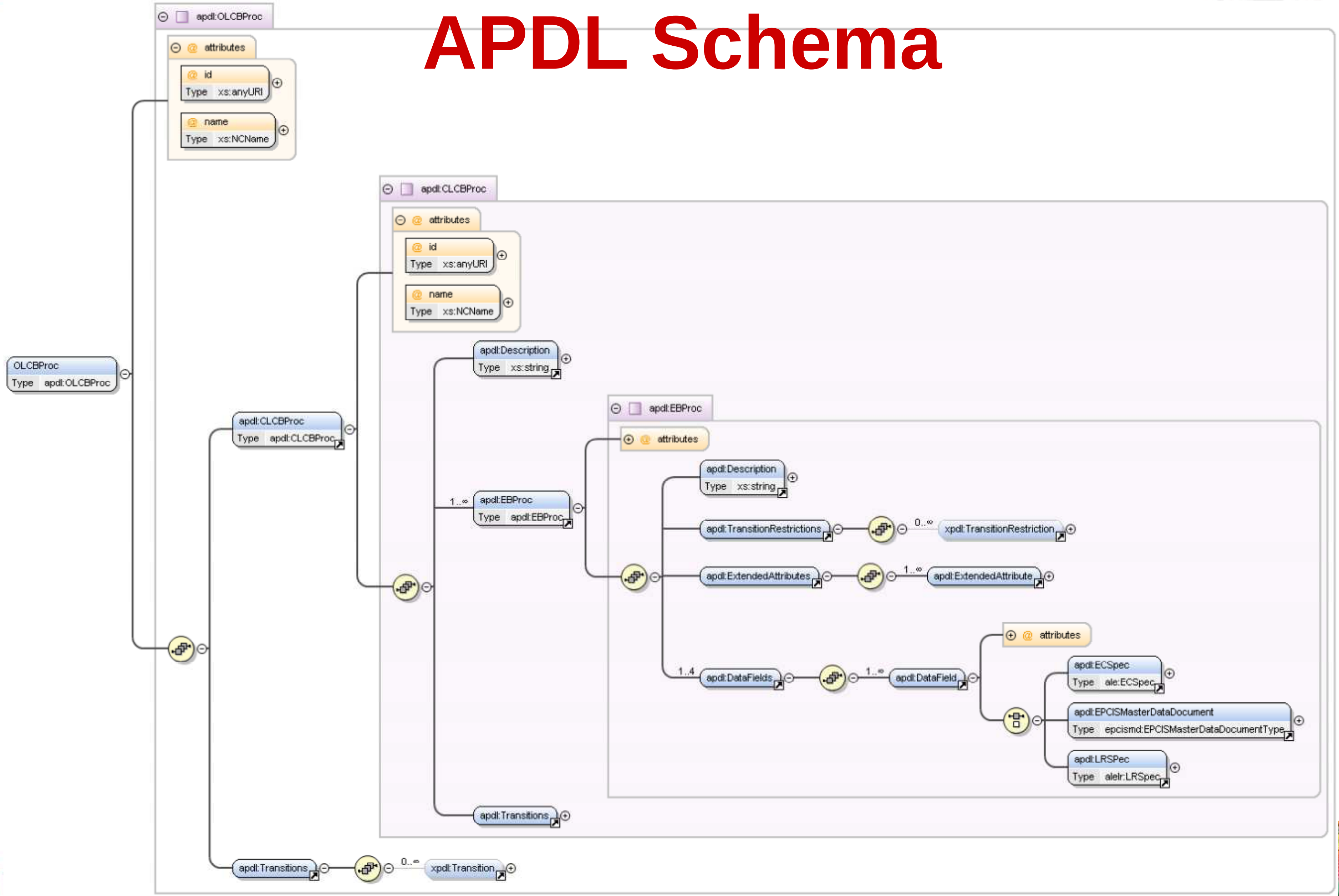
Programmable Meta-Language Structure (12)

- The Transitions Element

```
<xs:element name="Transitions">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="xpdl:Transition" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- Relates Elementary Business processes

APDL Schema


apdl:OLCBProc
attributes
id
Type xs:anyURI
name
Type xs:NCName
OLCBProc
Type apdl:OLCBProc
apdl:CLCBProc
attributes
id
Type xs:anyURI
name
Type xs:NCName
apdl:Description
Type xs:string
apdl:CLCBProc
Type apdl:CLCBProc
apdl:EBProc
attributes
apdl:Description
Type xs:string
1..∞
apdl:EBProc
Type apdl:EBProc
apdl:TransitionRestrictions
0..∞
xpdl:TransitionRestriction
apdl:ExtendedAttributes
1..∞
apdl:ExtendedAttribute
attributes
apdl:ECSpec
Type ale:ECSpec
1..4
apdl:DataFields
1..∞
apdl:DataField
apdl:EPCISMasterDataDocument
Type epcismd:EPCISMasterDataDocumentType
apdl:LRSPec
Type alelr:LRSpec
apdl:Transitions
apdl:Transitions
0..∞
xpdl:Transition

Describing Workflow Processes with APDL (1)

- ACME company
  - Owns a Central building and 3 Warehouses
  - Places orders for specific CPUs to Microchip Manufacturer
  - Needs a way to receive goods at Warehouse 1, Station 1 and inform its WMS

Describing Workflow Processes with APDL (2)

- Build a CBCProc Object

```
<OLCBProc>
    <!-- AspireRFID Process Description (Language Specification) -->

    <CLCBProc id="urn:epcglobal:fmcg:bti:acmesupplying"
    name="CompositeBusinessProcessName">
        <!-- RFID Composite Business Process Specification (the ID will be the
        Described Transactions's URI)-->
        <Description>Acme Supply Chain</Description>

        <EBProc Id="CLCBProcEnd" Name="CLCBProcEnd">
        </EBProc>

        <EBProc Id="CLCBProcStart" Name="CLCBProcStart">
        </EBProc>

        <EBProc id="urn:epcglobal:fmcg:bte:acmewarehouse1receive"
          name="AcmeWarehouse3Ship">
        </EBProc>

        <Transitions>
        </Transitions>
     </CLCBProc>

</OLCBProc>
```

Describing Workflow Processes with APDL (3)

- Build an EBCProc

```
<EBProc id="urn:epcglobal:fmcg:bte:acmewarehouse1receive" name="AcmeWarehouse3Ship">
    <!-- Elementary RFID Business Process Specification (the ID will be the
        Described Event's URI)-->
    <Description>Acme Warehouse 3 Receiving ReadPoint5 Gate3</Description>
    <TransitionRestrictions>
        <TransitionRestriction>
            <Join Type="AND"/>
        </TransitionRestriction>
    </TransitionRestrictions>
    <ExtendedAttributes>
        <ExtendedAttribute Name="XOffset" Value="204"/>
        <ExtendedAttribute Name="YOffset" Value="204"/>
        <ExtendedAttribute Name="CellHeight" Value="30"/>
        <ExtendedAttribute Name="CellWidth" Value="313"/>
        <ExtendedAttribute Name="ECSpecSubscriptionURI" Value="http://localhost:9999"/>
        <ExtendedAttribute Name="DefinedECSpecName"
    Value="Warehouse3RecievingObjectEvent"/>
        <!-- The DefinedLRSpecNames can be collected from the defined
     logicalReaders names at the ECSpec -->
        <!-- For the BEG configuration the port can be collected from the
    "ECSpecSubscriptionURI" value
    and the event to serve from the EBPSpec id -->
    </ExtendedAttributes>
    <DataFields>
    </DataFields>
</EBProc>
```

Describing Workflow Processes with APDL (4)

- F&C Module Data Fields: ECSpec

```
<DataField id="urn:epcglobal:fmcg:bte:acmewarehouse1receive_ecspec"
    type="ECSpec" name="RecievingECSpec">
    <ECSpec includeSpecInReports="false">
    <logicalReaders>
        <logicalReader>SmartLabImpinjSpeedwayLogicalReader</logicalReader>
    </logicalReaders>
    <boundarySpec>
        <repeatPeriod unit="MS">4500</repeatPeriod>
        <duration unit="MS">4500</duration>
        <stableSetInterval unit="MS">0</stableSetInterval>
    </boundarySpec>
    <reportSpecs>
        <reportSpec reportOnlyOnChange="false" reportName="bizTransactionIDs"
    reportIfEmpty="true">
        <reportSet set="CURRENT"/>
        <filterSpec>
            <includePatterns>
                <includePattern>urn:epc:pat:gid-96:145.12.*</includePattern>
            </includePatterns>
           <excludePatterns/>
        </filterSpec>
        <groupSpec/>
        <output includeTag="true" includeRawHex="true"
…………
```

Describing Workflow Processes with APDL (5)

- F&C Module Data Fields: ECSpec (cont'd)

```
………
                  <output includeTag="true" includeRawHex="true" includeRawDecimal="true"
              includeEPC="true“ includeCount="true"/>
                </reportSpec>
                <reportSpec reportOnlyOnChange="false" reportName="transactionItems"
reportIfEmpty="true">
                <reportSet set="ADDITIONS"/>
                <filterSpec>
                    <includePatterns>
        <includePattern>urn:epc:pat:gid-96:145.233.*</includePattern>
        <includePattern>urn:epc:pat:gid-96:145.255.*</includePattern>
            </includePatterns>
            <excludePatterns/>
        </filterSpec>
                <groupSpec/>
                <output includeTag="true" includeRawHex="true" includeRawDecimal="true"
        includeEPC="true“ includeCount="true"/>
                </reportSpec>
            </reportSpecs>
        <extension/>
    </ECSpec>
</DataField>
```

Describing Workflow Processes with APDL (6)

- F&C Module Data Fields: LRSpec

```
    <DataField
        id="urn:epcglobal:fmcg:bte:acmewarehouse1receive_lrspec"
        type="LRSpec" name=" SmartLabImpinjSpeedwayLogicalReader">
            <LRSPec>
                <isComposite>false</isComposite>
                <readers/>
                <properties>
                    <property>
                        <name>Description</name>
                        <value>
                        This Logical Reader consists of read point 1,2,3
                        </value>
                    </property>
                    <property>
                        <name>ConnectionPointAddress</name>
                        <value>192.168.212.238</value>
                    </property>
                    <property>
                        <name>ConnectionPointPort</name>
                        <value>5084</value>
                    </property>
………
```

Describing Workflow Processes with APDL (7)

- F&C Module Data Fields: LRSpec (cont'd)

```
………
                    <property>
                        <name>ReadTimeInterval</name>
                        <value>1000</value>
                    </property>
                    </property>
                    <property>
                        <name>PhysicalReaderSource</name>
                        <value>1,2,3</value>
                    </property>
                    <property>
                        <name>RoSpecID</name>
                        <value>1</value>
                    </property>
                    <property>
                        <name>ReaderType</name>
                        <value>
                    org.ow2.aspirerfid.ale.server.readers.llrp.LLRPAdaptor
                        </value>
                    </property>
                </properties>
            </LRSPec>
        </DataField>
```

Describing Workflow Processes with APDL (8)

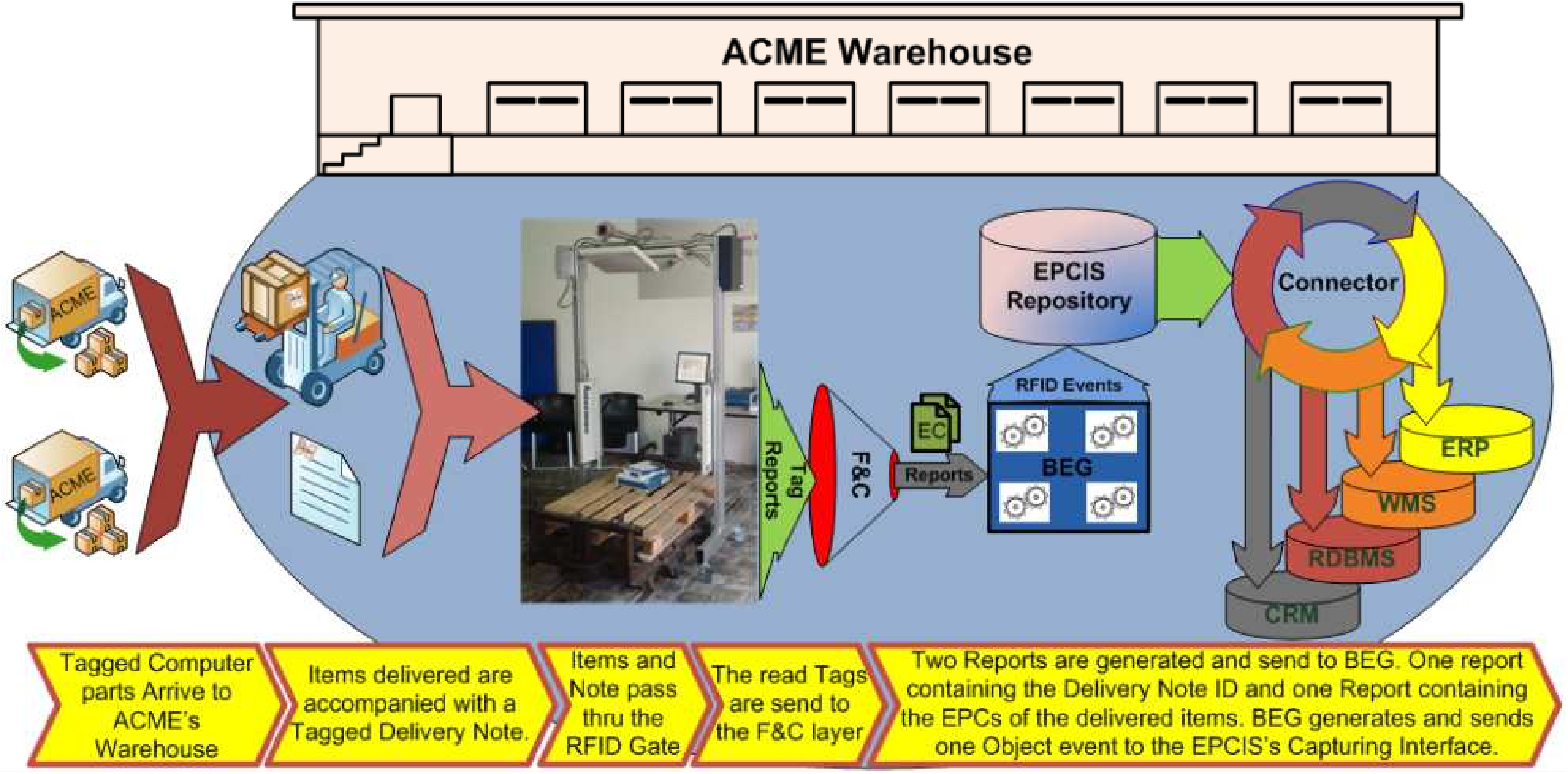
- BEG Module Data Fields: EPCISMasterDataDocument

```
<DataField id="urn:epcglobal:fmcg:bte:acmewarehouse1receive_masterdata"
type="EPCISMasterDataDocument" name="RecievingMasterData">
        <EPCISMasterDataDocument>
            <EPCISBody>
                <VocabularyList>
                    <Vocabulary type="urn:epcglobal:epcis:vtype:BusinessTransaction">
                        <VocabularyElementList>
                            <VocabularyElement id=" urn:epcglobal:fmcg:bte:acmewarehouse1receive">

    <attribute id="urn:epcglobal:epcis:mda:event_name">Warehouse1DocDoorReceive</attribute>
    <attribute id="urn:epcglobal:epcis:mda:event_type">ObjectEvent</attribute>
    <attribute id="urn:epcglobal:epcis:mda:business_step">urn:epcglobal:fmcg:bizstep:receiving</attribute>
    <attribute id="urn:epcglobal:epcis:mda:business_location">urn:epcglobal:fmcg:loc:acme:warehouse1</attribute>
    <attribute id="urn:epcglobal:epcis:mda:disposition">urn:epcglobal:fmcg:disp:in_progress</attribute>
    <attribute id="urn:epcglobal:epcis:mda:read_point">urn:epcglobal:fmcg:loc:rp:warehouse1docdoor</attribute>
    <attribute id="urn:epcglobal:epcis:mda:transaction_type">urn:epcglobal:fmcg:btt:receiving</attribute>
    <attribute id="urn:epcglobal:epcis:mda:action">OBSERVE</attribute>

                            </VocabularyElement>
                        </VocabularyElementList>
                    </Vocabulary>
                </VocabularyList>
            </EPCISBody>
        </EPCISMasterDataDocument>
    </DataField>
```

Describing Workflow Processes with APDL (9)

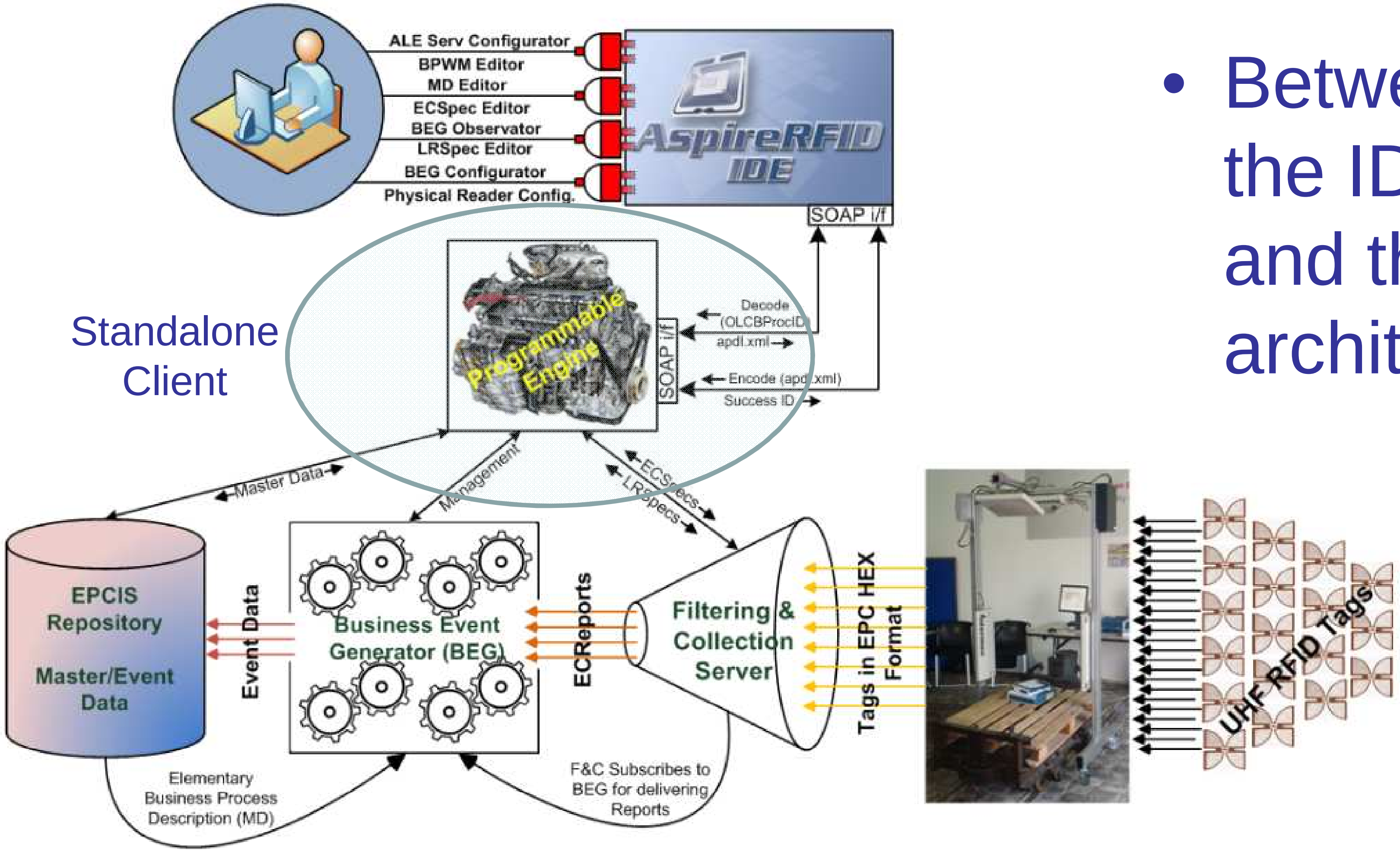

ACME Warehouse
ACME
ACME
Tag Reports
F&C
EC
Reports
RFID Events
BEG
EPCIS Repository
Connector
ERP
WMS
RDBMS
CRM
Tagged Computer parts Arrive to ACME's Warehouse
Items delivered are accompanied with a Tagged Delivery Note.
Items and Note pass thru the RFID Gate
The read Tags are send to the F&C layer
Two Reports are generated and send to BEG. One report containing the Delivery Note ID and one Report containing the EPCs of the delivered items. BEG generates and sends one Object event to the EPCIS's Capturing Interface.

ASPIRE Programmable Engine (APE)

ASPIRE Programmable Engine

- A run-time middleware module
  - Bridges APDL with the underlying middleware infrastructure
  - Uses APDL to
    - Encode (or program) the AspireRfid middleware
    - Decode, by retrieving all related information
  - Also, “hides” lower-level details

APE Role (1)



- Between the IDE and the architecture

APE Role (2)

- Configures the 3 important modules
  - The F&C Module
  - The BEG
  - The EPCIS
- Uses specific and defined interfaces

APE Role (3)

- Fully Based on SOA
  - Reveals 2 interfaces
    - Encode
    - Decode
  - Uses SOAP for message exchange

APE Interfaces (1)

- ALE Client API

Service Name	Input	Output	Info
define	specName : String spec : ECSpec	void	Creates a new ECSpec having the name specName, according to spec
getECSpecNames	-	List<String>	Returns an unordered list of the names of all ECSpecs that are visible to the caller
undefine	specName : String	void	Removes the ECSpec named specName that was previously created by the define method
subscribe	specName : String notificationURI : String	void	Adds a subscriber having the specified notificationURI to the set of current subscribers of the ECSpec named specName
unsubscribe	specName : String notificationURI : String	void	Removes a subscriber having the specified notificationURI from the set of current subscribers of the ECSpec named specName

APE Interfaces (2)

- ALE-LR Client API

Service Name	Input	Output	Info
getLogicalReader Names	-	List<String>	Returns an unordered list of the names of all logical readers that are visible to the caller. This list SHALL include both composite readers and base readers.
define	name : String	void	Creates a new logical reader named name according to spec
update	name : String	void	Changes the definition of the logical reader named name to match the specification in the spec parameter

APE Interfaces (3)

- BEG Client API

Service Name	Input	Output	Info
getEpcListForEvent	eventID: String	EventStatus, consists of : transactionID: String (denotes the Transactions ID) epcList: ArrayList<String> (stores all the read tags that are connected with the abovementioned Transaction)	Returns what is currently happening for a specific transaction
stopBegForEven	eventID: Sring	boolean	Stop serving a specific Event (described at the Master Data)
getStartedEvents	-	List<String>	Get all the Event IDs that are currently been served from the BEG
startBegForEvent	VocElem: VocabularyElementType[8] repositoryCaptureURL: String begListeningPort: String	boolean	Start a specific Event that is available at the EPCIS's Master Data
getEventList	repositoryQueryURL: String	List <VocabularyElementType>	Get all the Available Events (ready to be served) from the EPCIS's repository Master Data

APE Interfaces (4): Encode

- Encode API

Service Name	Input	Output	Info
encode	openLoopCBProc: OLCBProc	result: Integer	This method configures the AspireRFID middleware to serve the described Business Processes from the given APDL XML document. If the encode is successful the reply ID will be “400” if not the reply ID will be “425”.

- Inputs an XPDL XML file
- Returns execution code

APE Interfaces (5): Decode

- Decode API

Service Name	Input	Output	Info
decode	openLoopCBProcID: String	result: OLCBProc	This method returns an OLCBProc Object which is retrieved from an AspireRFID middleware running instance by a prior configured (encoded) OLCBProc by giving that object’s ID.

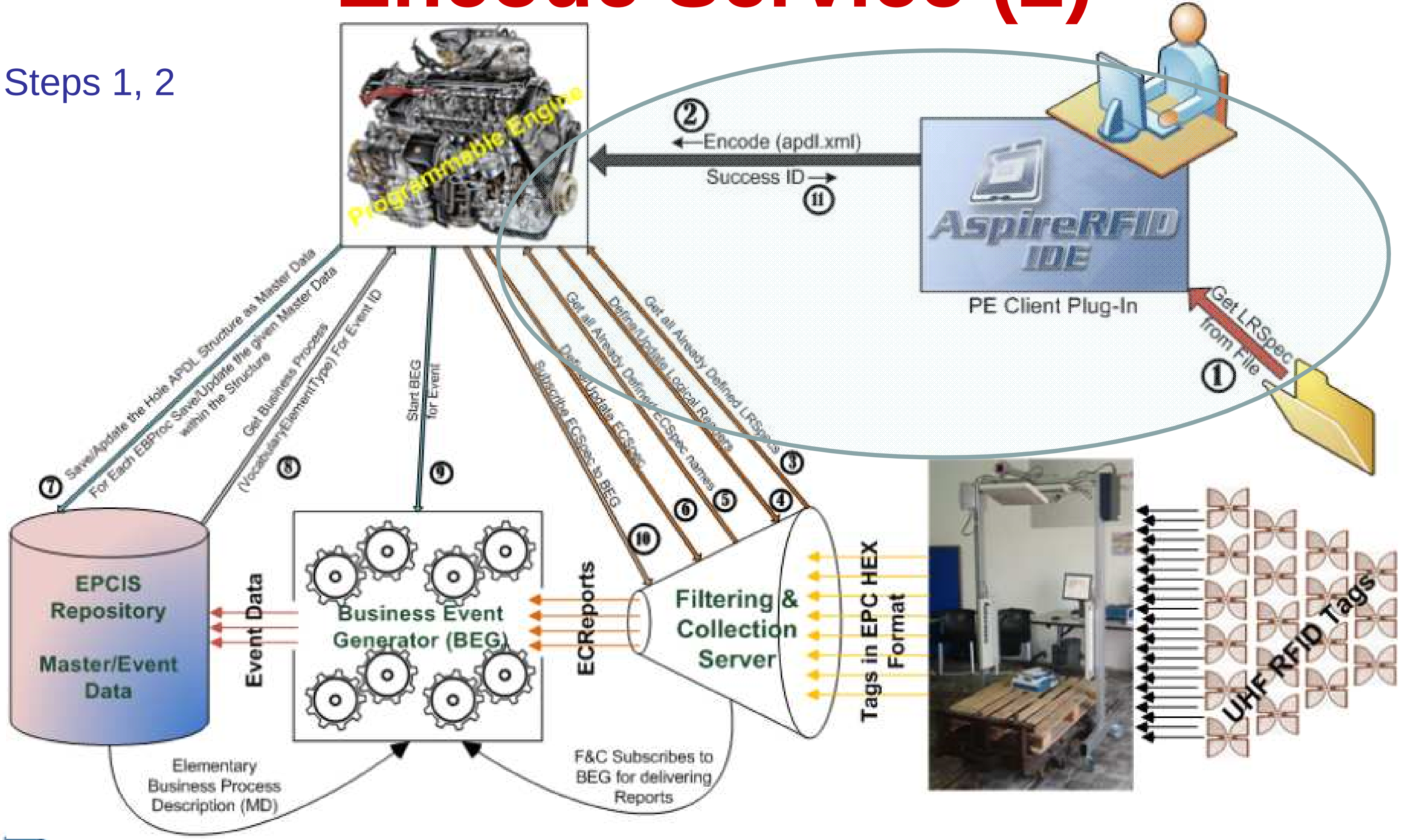
- Inputs an OLCBProc ID
- Returns the corresponding OLCBProc object

Encode Service (1)

- The PE Client
  - Retrieves LRSpec from file (step 1)
  - Maps it into an OLCBProc
    - using JAXB for XML binding
  - Delivers it to the PE server interface (step 2)
    - Using Apache CXF as a Web Service framework
  - The PE takes care of the rest!

Encode Service (2)

Steps 1, 2

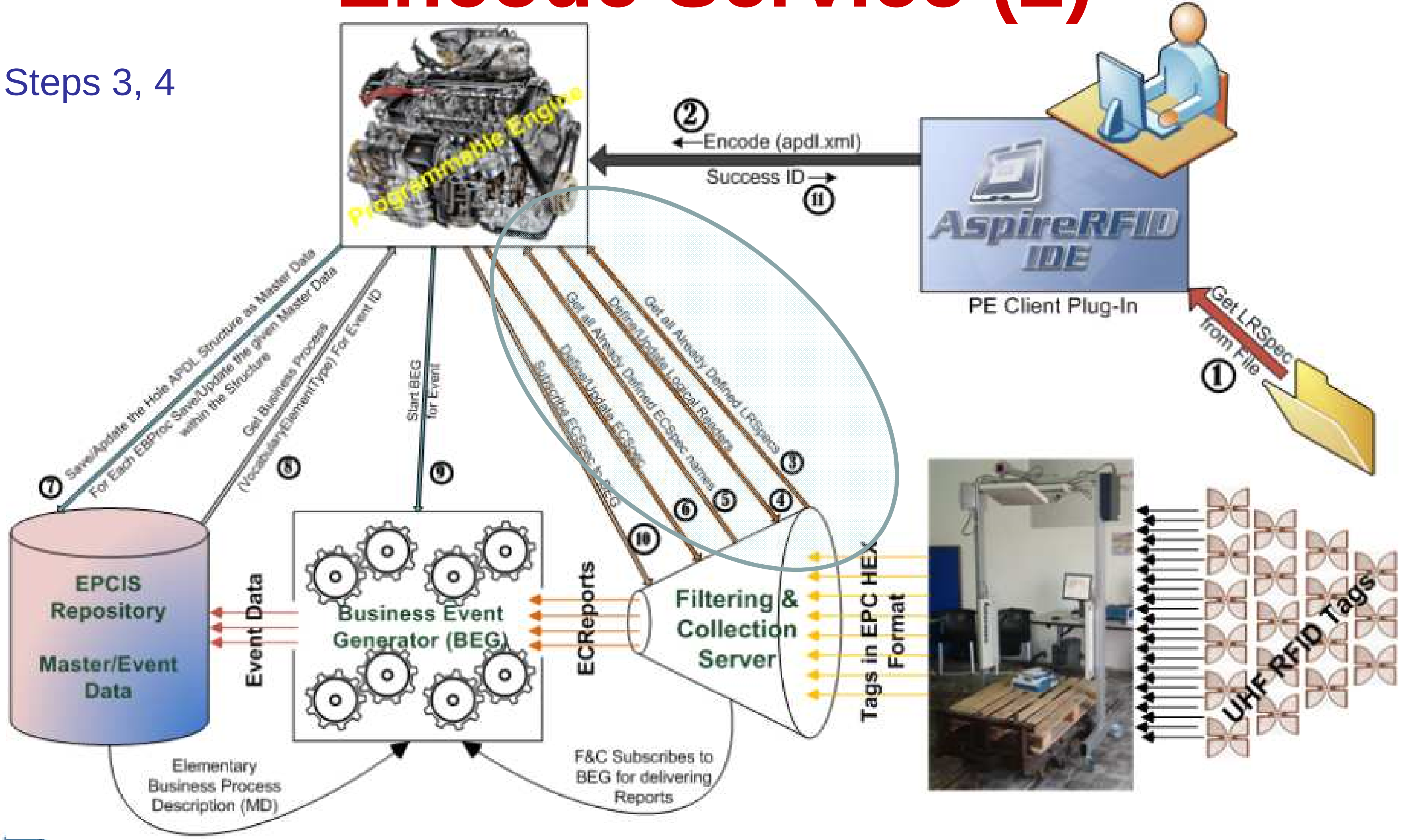

Programmable Engine
②
Encode (apdl.xml)
Success ID
⑪
AspireRFID IDE
PE Client Plug-In
Get LRSpec from File
①
Get all Already Defined LRSpecs
③
Define/Update Logical Readers
④
Get all Already Defined ECSpec names
⑤
Define/Update ECSpec
⑥
Subscribe ECSpec to BEG
⑩
Save/Update the Hole APDL Structure as Master Data
For Each EBProc Save/Update the given Master Data within the Structure
⑦
Get Business Process (VocabularyElementType) For Event ID
⑧
Start BEG for Event
⑨
EPCIS Repository
Master/Event Data
Event Data
Business Event Generator (BEG)
ECReports
Filtering & Collection Server
Tags in EPC HEX Format
UHF RFID Tags
Elementary Business Process Description (MD)
F&C Subscribes to BEG for delivering Reports

Encode Service (3)

- The PE Server
  - Analyzes OLCBProc into
    - CLCBProc's
    - EBProc's
  - Creates ProcessedEBProc object
  - Setups ALE-LR
    - Get Reader names (step 3)
    - Define/Update readers (step 4)

Encode Service (2)

Steps 3, 4

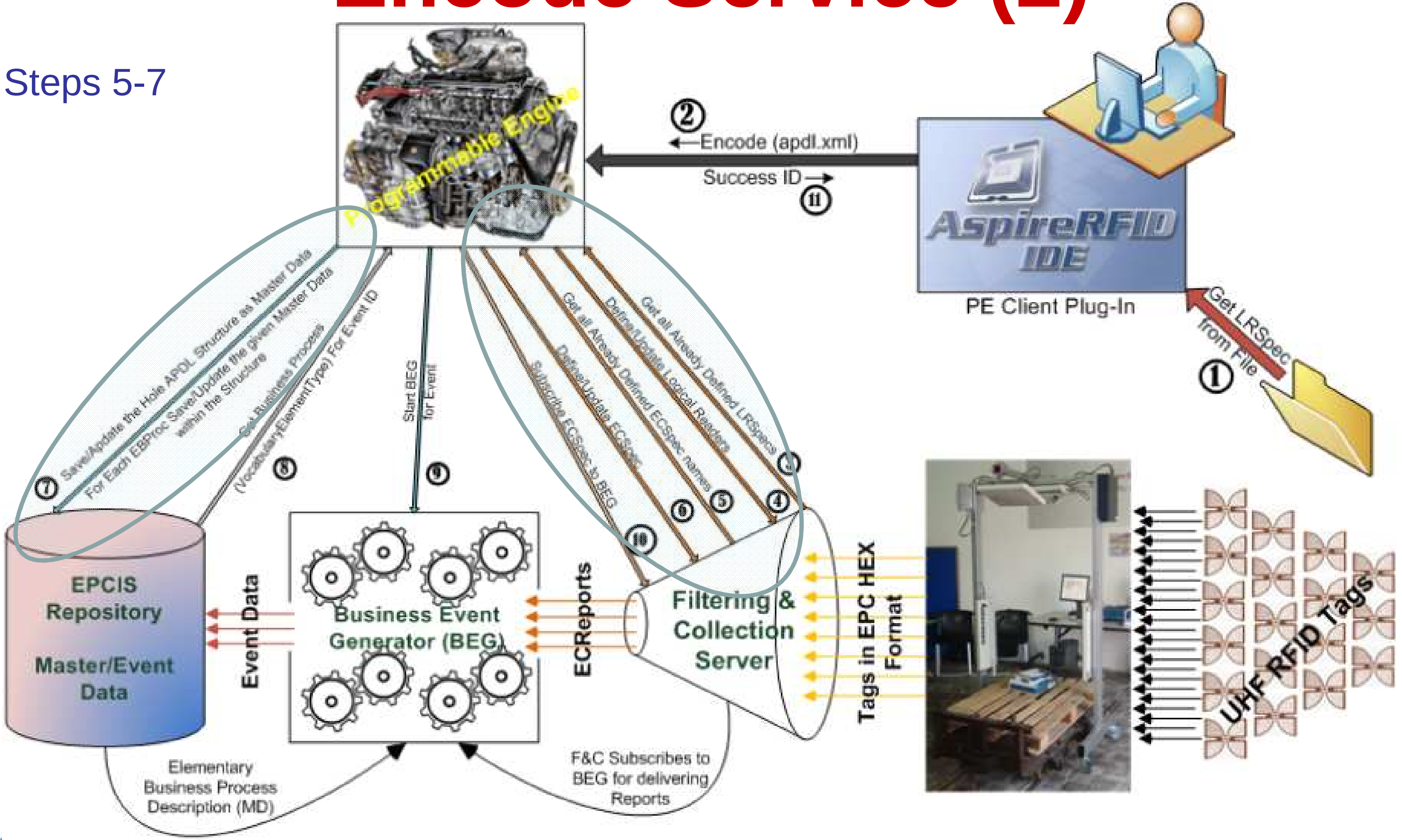

Programmable Engine
② Encode (apdl.xml)
Success ID ⑪
AspireRFID IDE
PE Client Plug-In
Get LRSpec from File ①
Get all Already Defined LRSpecs ③
Define/Update Logical Readers ④
Get all Already Defined ECSpec names ⑤
Define/Update ECSpec ⑥
Subscribe ECSpec to BEG ⑩
⑦ Save/Update the Hole APDL Structure as Master Data
For Each EBProc Save/Update the given Master Data within the Structure
Get Business Process (VocabularyElementType) For Event ID ⑧
Start BEG for Event ⑨
EPCIS Repository
Master/Event Data
Event Data
Business Event Generator (BEG)
ECReports
Filtering & Collection Server
Tags in EPC HEX Format
UHF RFID Tags
Elementary Business Process Description (MD)
F&C Subscribes to BEG for delivering Reports

Encode Service (4)

- The PE Server
  - – Configures ALE
    - Get the ECSpec names (step 5)
    - Define/Update ECSpec's (step 6)
  - – Configures EPCIS
    - Using the EPCIS Capture interface (step 7)

Encode Service (2)

Steps 5-7

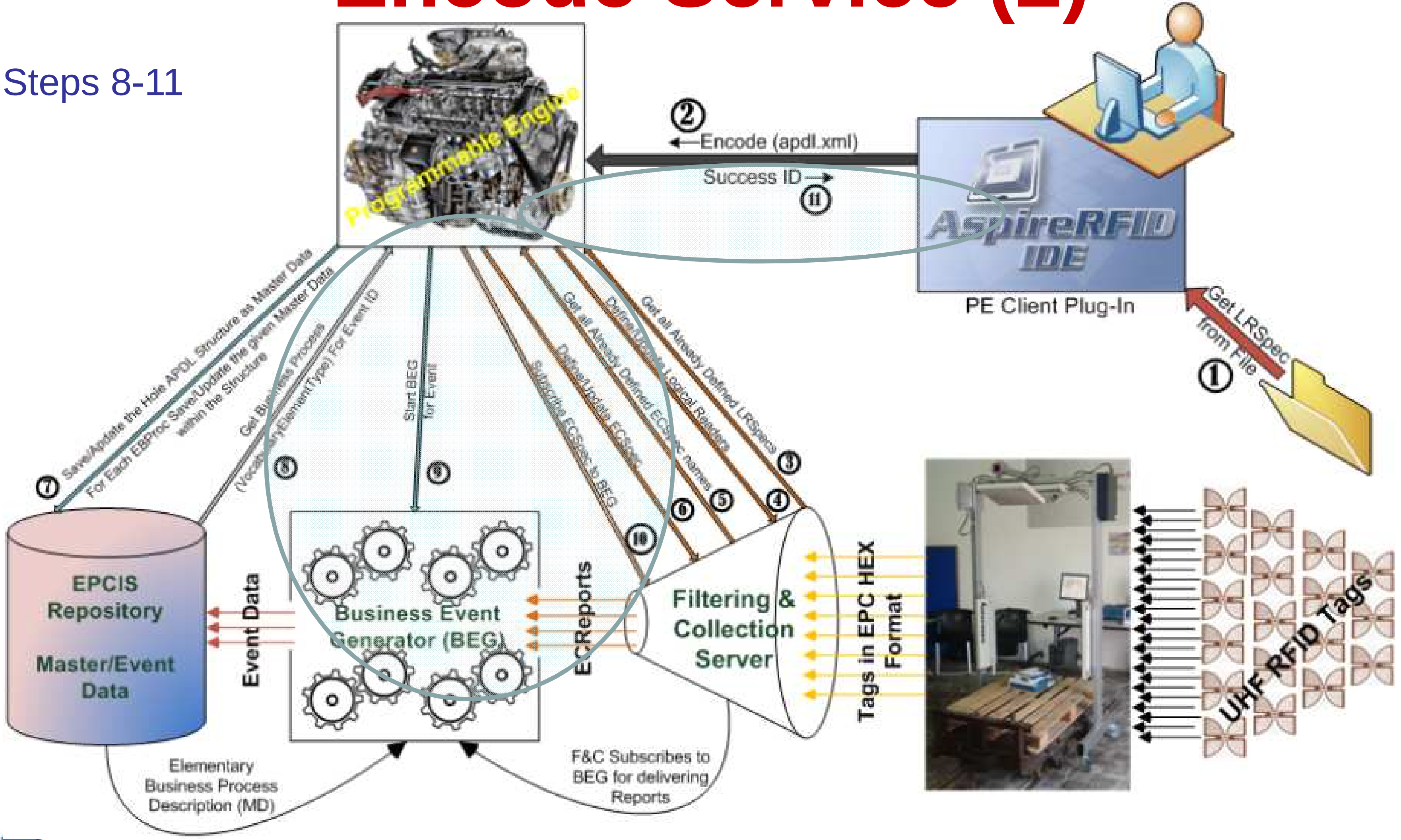

Programmable Engine
② Encode (apdl.xml)
Success ID ⑪
AspireRFID IDE
PE Client Plug-In
Get LRSpec from File ①
⑦ Save/Update the Hole APDL Structure as Master Data
For Each EBProc Save/Update the given Master Data within the Structure
Get Business Process (VocabularyElementType) For Event ID ⑧
Start BEG for Event ⑨
Get all Already Defined LRSpecs ③
Define/Update Logical Readers ④
Get all Already Defined ECSpec names ⑤
Define/Update ECSpec ⑥
Subscribe ECSpec to BEG ⑩
EPCIS Repository
Master/Event Data
Event Data
Business Event Generator (BEG)
ECReports
Filtering & Collection Server
Tags in EPC HEX Format
UHF RFID Tags
Elementary Business Process Description (MD)
F&C Subscribes to BEG for delivering Reports

Encode Service (5)

- The PE Server
  - Configures BEG
    - Get the VocabularyElementType from EPCIS for each specific EBProc ID (step 8)
    - Configure BEG's functionality for the given EBProc ID (steps 9, 10)
  - Sends back to the client the execution code (step 11)
    - 400 for success
    - 425 for failure

Encode Service (2)

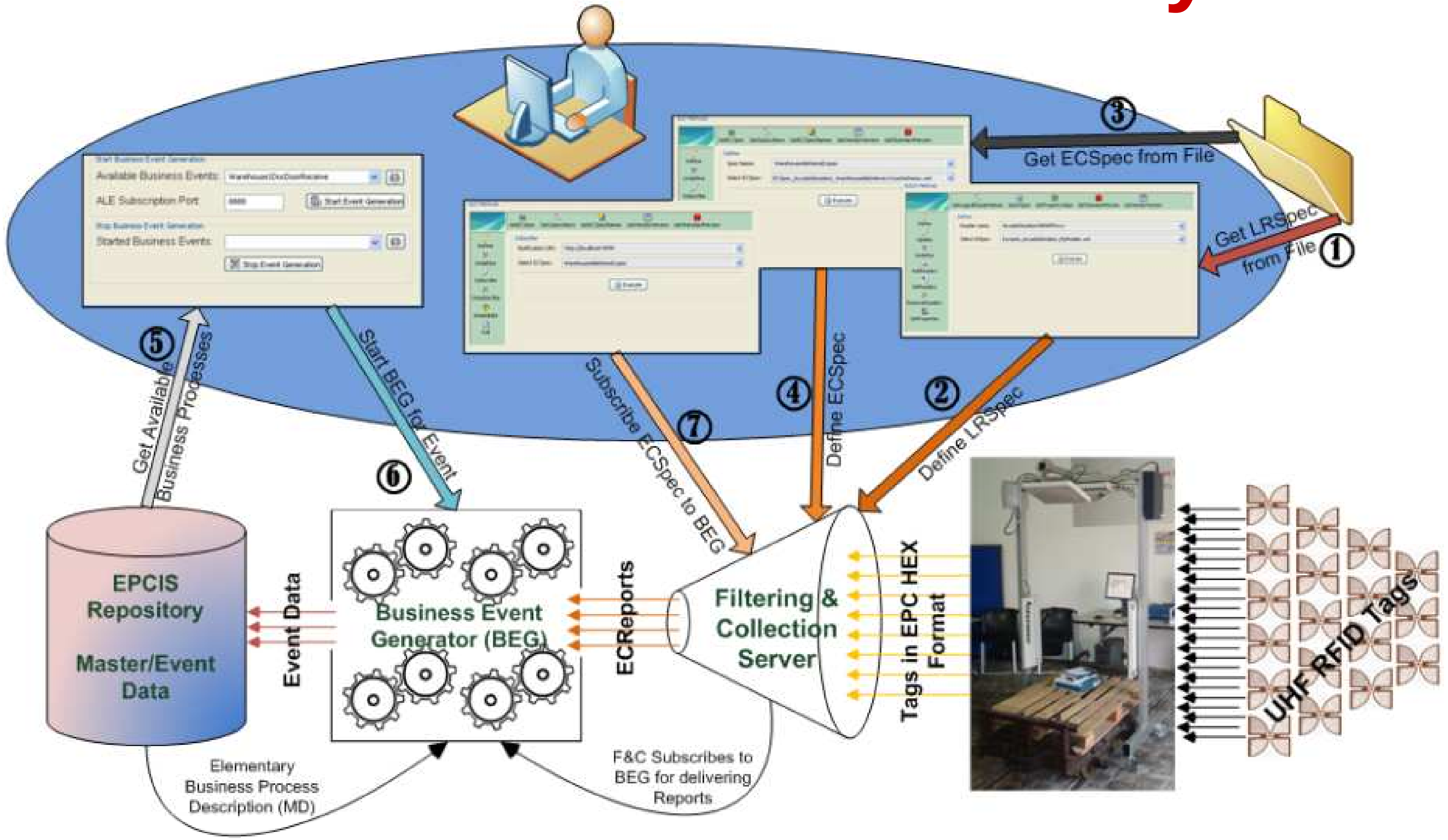
Steps 8-11


Programmable Engine
② Encode (apdl.xml)
Success ID ⑪
AspireRFID IDE
PE Client Plug-In
Get LRSpec from File ①
⑦ Save/Update the Hole APDL Structure as Master Data
For Each EBProc Save/Update the given Master Data within the Structure
Get Business Process (VocabularyElement(Type)) For Event ID ⑧
Start BEG for Event ⑨
Subscribe ECSpec to BEG ⑩
Define/Update ECSpec ⑥
Get all Already Defined ECSpec names ⑤
Define/Update Logical Readers ④
Get all Already Defined LRSpecs ③
EPCIS Repository
Master/Event Data
Event Data
Business Event Generator (BEG)
ECReports
Filtering & Collection Server
Tags in EPC HEX Format
UHF RFID Tags
Elementary Business Process Description (MD)
F&C Subscribes to BEG for delivering Reports

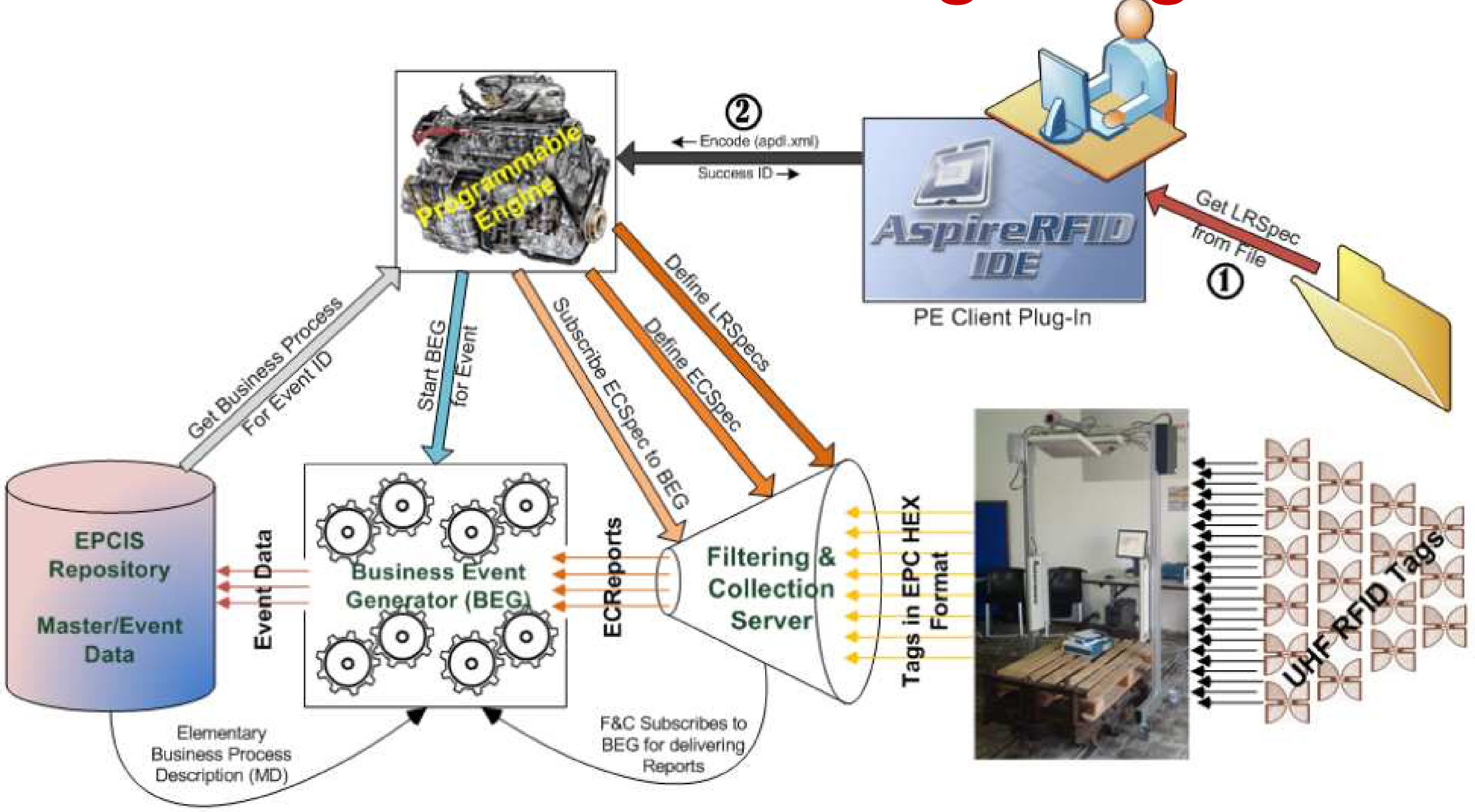
Configuring AspireRfid

- The Conventional Way
  - (Seven steps) x (Number of Elementary Business Processes)
- PE-based
  - Two steps
    - Retrieve apdl.xml
    - Encode it using the PE Client

The Conventional Way



PE-Based Configuring



Conclusions

- Overview of Existing BPM Languages
- Designed and implemented APDL
  - Based implementation on XPDL
- Implemented ASPIRE Programmable Engine that
  - Uses APDL to
    - Encode and decode RFID solutions
    - Ease middleware programmability

References – Additional Reading

- ASPIRE Public Deliverable D4.2a
- ASPIRE Public Deliverable D4.3b
- ASPIRE Public Deliverable D4.4a
- M. Weske: Business Process Management: Concepts, Languages, Architectures, Springer (2007)
- Workflow Management Coalition Workflow Standard, “Workflow Process Definition Interface -- XML Process Definition Language V1.0”, Document Number WFMC-TC-1025, October 25 (2002)
- Aalst, W. M., Mulyar, S., Russell, N., & Arthur, H.: Workflow Control-Flow Patterns - A Revised View (2006)