SAP Architecture Blueprint

Version: 1.1

Date: 29.01.2008

Authors:

Thomas Mayer

Thomas Rumbach

Reinhard Brede

Joachim Münter

Next Generation AII/OER

Project Name/cProject Title:	**NextGen AII/OER**
Sponsor/Project Initiator:	**Krish Mantripragada**
Program/Project Lead:	**Reinhard Brede/Joachim Münter**
Lead Architect:	**Thomas Mayer NG AII**
	Thomas Rumbach, NG OER
Development:	☒ SAP AG Walldorf
	☐ Partner/ISV

Document History

Version	Date	Status (Comments)
1.0	2007-12-05	Review Version
1.1	2008-01-29	Released Version

I MARKET AND PRODUCT BACKGROUND OF PROJECT/PROGRAM

Planned release date:	2007-2008 Prototype for architecture and technical platform evaluation 2008-2009 Product development ~2010 Shipment	
Underlying SAP NetWeaver release:	NetWeaver CE for design and run time of NG AII and NG OER application NetWeaver 04s for database persistence of both applications	
Used SAP NetWeaver stacks:	☒ ABAP	☒ J2EE/Java EE 5
Target platform (OS, DB):	All available via NetWeaver CE OER NetWeaver 7.0 But databases limited to those providing native XML support	

Targeted market segments / sub-segments:

Prototype for next generation of Radio Frequency Identification (RFID) enabling platform (NG AII) and object event repository (NG OER)

Use cases targeted by the project/program:

Next Generation Auto Identification Infrastructure (NG AII):

- Graphical modelling of business processes
- Configurable outbound logistics based on serialized parts with pack and load step
- Configurable inbound logistics with unload step based on serialized parts
- Goods movements between storage locations captured through fixed gates or mobile read devices
- Mixed deliveries comprising tagged, serialized and non-tagged parts

Next Generation Object Event Repository (NG OER):

- Event capturing and event query services compatible with the corresponding EPCIS services
- Configurable product authentication logic upon receiving events in inbound and outbound processes
- Flexible User Interface to display content of the OER database to support ID details and document details monitoring

Strategic goals SAP wants to achieve with the project/program:

Provide new Real World Awareness (RWA) platform supporting the following two main functions:

- NG AII:

  To enable automatic close real time recording of serialized items comprising a graphical modelling tool and a lean deployable runtime. Efficient scaling-up through multiple instances

- NG OER:

  The term “Object Event Repository” comprises the functions to process and store business events. Flexible configuration of the data structure of the business events and the rules to process these events at design time is essential. Typical applications supported by the event processing rules are tracking, alerting and analytics

Mandatory software capabilities to address goals, use cases, and target market:

- NG AII:
  - Easily and remotely deployable lightweight runtime
  - Scalable for high performance and mass volume Auto ID (e.g. RFID) business
  - Meet flexibility requirements for modelling and graphical modelling of Auto ID enabled business processes
  - Meet UI requirements for shop floor environment (big letters and buttons, touch screen, push refresh, response time between receiving reader message and UI refresh ≤ 1 sec)
- NG OER:
  - High throughput while updating the event database with new event records (> 100 event records per second containing ~30 IDs of tracked objects in average)
  - High query load in parallel to the high updating load (> 100 simple queries (i.e. selects by ID) per second)
  - Very high capacity of the database (> 100 TBytes of event data => compression function required)
  - Special services to provide object (entity) relationship information (e.g. packaging hierarchy or IDs relating to a business document)
  - Configurable event processing and alerting upon event capturing
  - Integration of event processing with the backend systems

II. ARCHITECTURE

Preface

This document describes the software architecture for the project NextGen AII / OER, which is part of the program Real World Awareness (RWA). “Real World Awareness” in this context is the ability to sense information from any type of object (physical objects, documents, people, IT systems etc.) in real or near real time and to respond accordingly depending on the business context where the sensing event occurred. It is especially related to techniques like RFID, which provide a means of tracking information on a level of detail and over the whole life cycle of the labeled object in situations where this was not manageable so far for practical reasons (e.g. serial number tracking of single pharmaceutical drug packages).

The goal of the Real World Awareness Program is to provide a platform to integrate the sensor information into our existing software solutions. The platform should provide means to model the (business) processes that are triggered by the sensing event in a flexible manner: It should provide a lean runtime to execute the modeled processes and rules. And it should provide a high-volume database which can efficiently be updated at a high throughput rate while concurrently processing a high load of queries. The need for databases that are capable to handle a new dimension of volume and a new dimension of flexibility regarding the structure of the stored information is a consequence of the ability of the new RWA techniques to efficiently acquire information on a detailed (mainly item related) level.

The Real World Awareness Platform has two central usage types, the “Auto-ID Infrastructure” (AII) and the “Object Event Repository” (OER):

The Auto-ID Infrastructure comprises all functions, which are close to the sensing device (RFID reader, active RFID transponder, 2D barcode reader etc.). It is used to enrich the information that is sent by the reader (i.e. mainly the unique item ID) by information about the business context in which the read was executed. When reading the ID of a pallet, for instance, the AII is the instance to “know” that the read was to post a goods issue and it adds the information for which delivery the goods issue was posted. The document that describes the information sent by the reader and its business context is called an “event”. In a more general sense the AII is not only used to create events but also to monitor and control logistic transport process steps (e.g. pick, pack, load, unload) on a middle ware level. The fulfillment of these process steps is reported to the backend system (usually ERP) and OER. To support the use of NG AII per process step, it has to collaborate closely with the backend system (e.g. in the case, that process steps are executed in parallel). Enterprise services should be used for the integration of the backend systems.

In usage type “OER” the emphasis is more on the database functions of the RWA platform and the infrastructure to collect and process the events generated by connected AII systems or equivalent external systems. The OER has two major building blocks, the event database (also called the event persistence) and the event processing. “Event processing” in this context means the business process, which are automatically triggered upon receiving an event in OER. If compared to the AII “event processing” in OER is more on a business process level (e.g. automatically trigger an alert if a tracked pallet is overdue), while event processing in AII is more on a detail and document level (e.g. find the delivery document for the item identified by the reader upon goods receipt). Similar in both cases is, that the rules for event processing are strongly implementation-specific so, that a tool for easy configuration is essential. The challenge for the design of the database is, that the structure of “events”, i.e. the event documents to be handled by a single OER instance has a broad variance while at the same time high throughput rates (> 10.000 Observations / second) and data volumes (> 100 TByte) have to be managed.

Main Architecture Concepts and Decisions (Runtime)

This paragraph describing the main architecture concepts of the RWA platform is divided into three sections: The first called “General concepts” describes features which are relevant for both usage types. The two following sections describe features that are relevant for the specific usage types AII and OER.

- General Concepts

  As pointed out in the preface chapter the most important common characteristic of all usage types of the RWA platform is “event processing”. In case of the AII usage type "events" in that sense are mainly the "read events" (e.g. RFID read events) while in case of the OER usage type the events are more "business events", i.e. (XML) documents that describe a business event that happened to one or objects identified in the document by a unique ID. From a tool perspective, however, the requirements to the configuration and modeling tool are more or less the same. Thus - as shown in the overall architecture picture below - Galaxy was selected as the modeling tool for both AII and OER.

  Any activity modeled in Galaxy describes on a technical level nothing else but a method call (currently a web service call, in the future EJB or local method call). The modeling describes the sequence in which and the conditions on which a method is called, plus the parameters passed to and returned from the method. Thus as a basis for the Galaxy modeling a set of pre-implemented EJBs and/or methods is needed as part of the “toolbox”. This method collection builds the “Service Layer” shown in the raw architecture picture. Typically some of the methods provide services that relate more specifically to either the AII or the OER usage type. But there are also services that are needed in both cases. Consequently the methods (services) in the service layer are grouped into the three categories “AII Related Service”, “OER Related Services” and “Commonly Used Services”. This grouping is, of course, more a logical grouping from a business perspective. Since the assignment of a service to one these groups is not unambiguous in many cases, it is, of course, possible to use any service from any group in each usage type.

- NG AII

  The business application NG AII consists of several Galaxy process models (meaning: process models of logistic process steps designed with the Galaxy Workbench), activities and functions. The activities and functions are implemented in Java to be usable in Galaxy. They are designed in a way to be reusable in other (customer) process models.

  The provided local services are defined thru the use cases of NG AII (see chapters below: Product Definition and Specifications) during delivery processing with the steps pick, pack, load and unload.

  Two main user interfaces are offered for this purpose:

  - The user interface of the supervisor (purpose: monitoring of the process steps and correction of the process step data) is implemented with Web Dynpro Java. The supervisor UI provides information over all currently active processes in an AII system instance.
  - The user interface of the worker (directly involved in the process step; provides fast feedback to the scanned objects/data) is implemented with Adobe Flex to provide push refresh, large text, and buttons and to support touch screens.

- NG OER

  As pointed out in the preface chapter the second central building block of the OER besides “event processing” is the Persistence Layer The Persistence Layer is used to store events (i.e. “business events” in the sense described above) that where either created by AII, or, by an external remote application. Events received from remote applications are received through the EPCIS compatible “Event Capture Interface”. This interface is placed in the Event Processing Layer (and not in the

Persistence Layer) because events coming from remote locations may not only be stored but can trigger the same or similar processing as events coming from attached local AII instances do.

One of the major requirements the architecture has to deal with is the flexibility of the data structure of the events, which is strongly influenced by the flexibility XML schemes provide.

On a first level the scope of so-called “event types” is relatively broad. This means a considerable variability of the data structures to represent a certain event type needs to be supported. In addition, it must be possible to add a new event type at any point in time while the system is running.

On a second level the variety of the data structures is even more increased by the fact that the data structure per event type is also not stable but contains so-called “extension points” that can take any number of parameters of any data type. These parameters can be different from event to event, and(!), must (at least for a limited number of parameters) be efficiently searchable.

This kind of flexibility must be provided while at the same time the requirements regarding data volume attain a level which is at least one order of magnitude higher than the numbers reached in typical DB installations (> 100TB uncompressed). The relatively high update rates, current query load, backup load and archiving have to be assured even if the database reaches this very high data volume. Therefore the decision was made, to evaluate the native XML features of databases like IBM DB2, Oracle or Microsoft SQL in the context of a prototype project. Regarding the overall architecture, however, it is important to note, that in case it turns out, that the XML approach is not suitable for the final product, the main building blocks and the interfaces of the event persistence stay the same. Only the internal data structures of the Persistence Layer will change in that case.

Besides storing the events the Persistence Layer is also used to store or access some other business object types, namely master data and analytical data for BI. These are described in more detail in the next section.

The architecture picture also describes that the Persistence Layer is implemented in ABAP while the remaining parts of the platform are implemented on NetWeaver CE in Java. The decision for Java is mainly driven by the fact that Galaxy is only available on this platform, but it also opens the opportunity to get a really lean run-time (e.g. OSGi). The decision for ABAP is induced by the experience that ABAP is more stable and more reliable for database and “heavy-weight” applications. Since this approach is less advantageous from a TCO and a performance perspective, implementing the Persistence Layer on NetWeaver CE, too, may be an alternative to be considered.

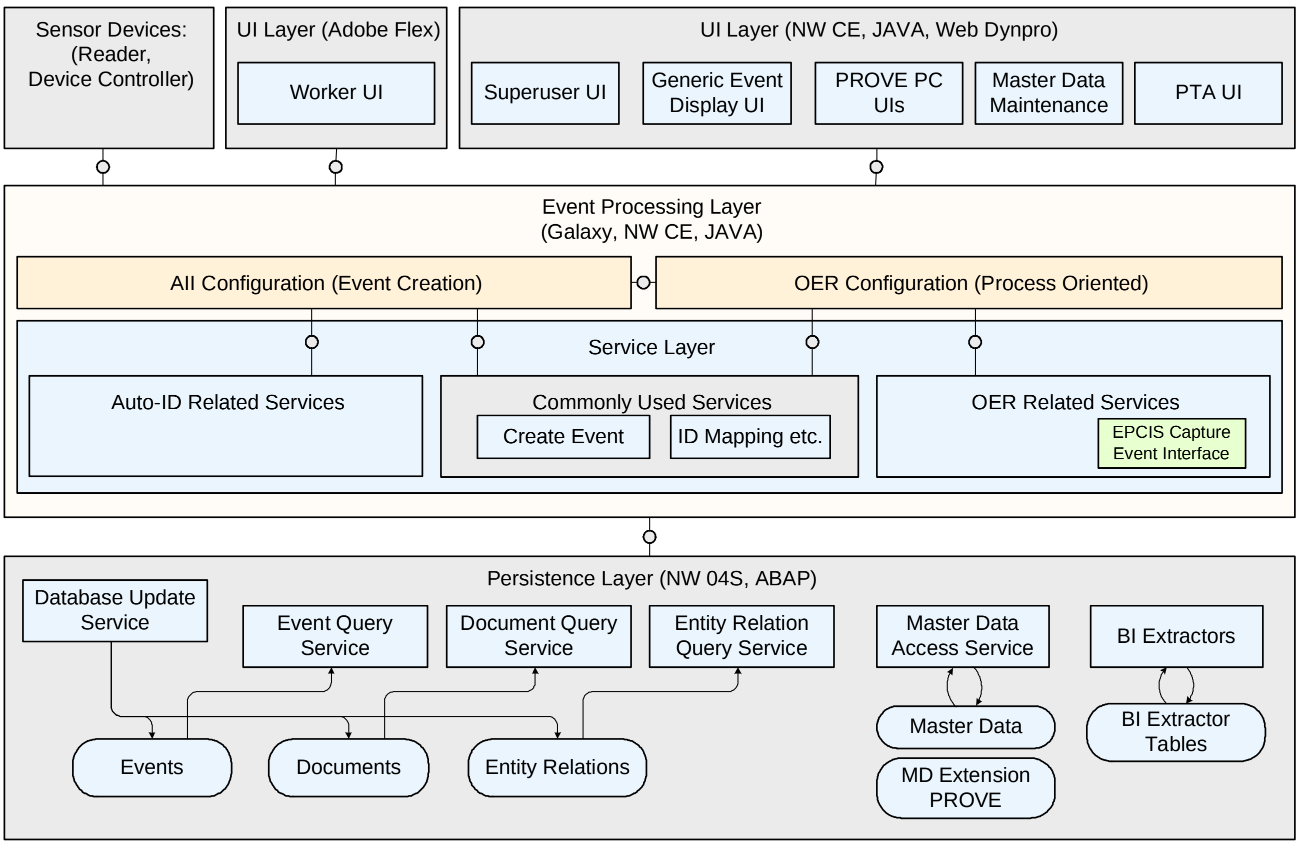


Figure 1: Architecture Overview Diagram

Main Architecture Building Blocks

Galaxy

The core of the application is Galaxy as the process engine. It will provide interfaces to different communication protocols via the Unified Integration Layer (registered requirements: communication via HTTP, SOAP, RFCs, IDocs w/ and w/o using XI).

The application itself exists out of several preconfigured Galaxy process models (based on the use cases defined for the prototype), a bunch of functions, and activities. The functions and activities (in the case they are no Web Service Calls to the backend system) are implemented in Java (Service Layer). The process models and the Java code are delivered and deployed together.

The functions and activities can be reused in the graphical designed process models (done in Galaxy Workbench). They are providing access to following services:

- Master data (e.g. material and business partner), including cache of this data
- Encoding/Decoding (Unique Identification including number ranges)
- Persistence Layer
- Read point descriptions (master data, necessary for AII)
- User interfaces

Commonly Used Services

Services that are commonly used by the AII usage type and the OER usage type are combined in the corresponding group of the Service Layer. Depending on the type of service they are implemented in this layer directly or accessing only ‘deeper’ layers (e.g. master data service in the Persistence Layer). If necessary also caching functions (e.g. of the master data) are provided in this layer. For the using application it is transparent, in which stack the main task of the service is fulfilled.

At the implementation it has to be insured, that any object is cached in one layer only. This should be as near as possible at the using layer to reduce the network traffic. This level of caching has to be considered in the models designed in Galaxy (to ensure, that the cached data don't go out of date).

Persistence Layer

The Persistence Layer is used to store all types of objects that are specific for the RWA platform. One of the basic principles behind the present architecture is the idea to replicate as few information as possible from the original systems into the RWA platform (ideally none at all). Instead the required information is retrieved from the source systems ad (where suitable for performance reasons cached in appropriate services of the RWA platform. The objects that are specific for RWA and thus the only to make persistent in the RWA Persistence Layer are:

- Events

  "Events" stored in the Persistence Layer are the "business events" mentioned earlier in this document to distinguish them from "read events" which is the raw information pushed from the read device (e.g. RFID reader) to the platform. These events are documents (typically in the present case XML documents) which describe when, where and in which business context an object or (more general) an entity was observed. Events may contain parameter lists that vary not only by event type but also by instance.

- Documents

  Documents in the sense of the Persistence Layer are documents that reference a business document, whose original is stored in the backend OLTP systems (e.g. ERP) and that contain a list of RWA specific attributes per referenced document. Attribute lists can exist on each level of the document structure, i.e. on header, item line, schedule line or further levels. Corresponding to the principle to replicate as few information as possible from the source systems to RWA the emphasis is on attributes that are "RWA specific". Replicated attribute should be an exception.

- Entity Relations

  Entity relations describe hierarchical relations of objects or entities to each other. Relations in that sense can describe the physical aggregation of objects, e.g. the packaging hierarchy (10 bottles in 1 box, 100 boxes on 1 pallet, and so on). They can also describe a "logical" relationship, e.g. a list of the serialized items in a specific delivery. Any relationship of entities represented by a unique ID to other entities represented by a unique ID is an entity relation in the sense of the Persistence Layer.

  The entity relationship information is extracted from the events and in that regards redundant information. It is extracted, however, upon receiving an event in the Persistence Layer to improve the performance of services that provide those hierarchical relationships for calling applications. The update is effected through fix programs within the Persistence Layer so, that the number of relation types is limited to the ones provide by SAP. It should not be freely configurable due to performance impacts.

- Master Data

  Master Data store in the Persistence Layer are those master data that are not available in the local backend systems but originate from external systems belonging to an arbitrary number of external partners. They are need for the OER usage type of the RWA platform, for instance, in case the OER is configured as EPCIS. Accessing the systems keeping the originals is not suitable in that case for performance and data quality reasons. Corresponding to use cases like EPCIS the data structure of the master data must be flexibly configurable in a similar way as this was described for the events.

- BI extractors

  The Persistence Layer must provide a configuration to define the structure of extractor tables for BI. Those extractor tables must pre-aggregate event information and information derived from evaluating a set of interrelated events, in order to reduce the update rate for BI to a rate BI can handle (one event must not generate one or several BI records, but the information from several (i.e. 50-100) events must be combined into one record for BI. The extractor tables are update through services in the Service Layer (not internal "hard-wired" update in the Persistence Layer.

As shown in the architecture picture above the event persistence has appropriate service interfaces to update and retrieve data of the types mentioned. A special case is the update of events, documents and entity relations. This update comes for all of these three business object types through a single interface and in a single transaction. This is to make sure that the information for these three object types which strongly relates to each other is always consistent.

NG AII Architecture

NG AII will collect several messages belonging to one process (e.g. packing of a delivery) and hand over only one event to the follow up systems (like OER and/or the backend system) if the process step is finished. This will reduce the number of messages and service calls, handled in the central systems. To reach the high volume and performance requirements

- Performance tuned Java implementations of the activities and functions and
- Caching mechanism of Java (master data) and Galaxy (process instances) will be used

Beside that further performance improvements will be possible by using several AII runtime instances (scalability). Any of these AII instances can use a central Persistence Layer together or have its own local Persistence Layer (instead of a central one). The central Persistent Layer will be used in this case for tracking and monitoring functions.

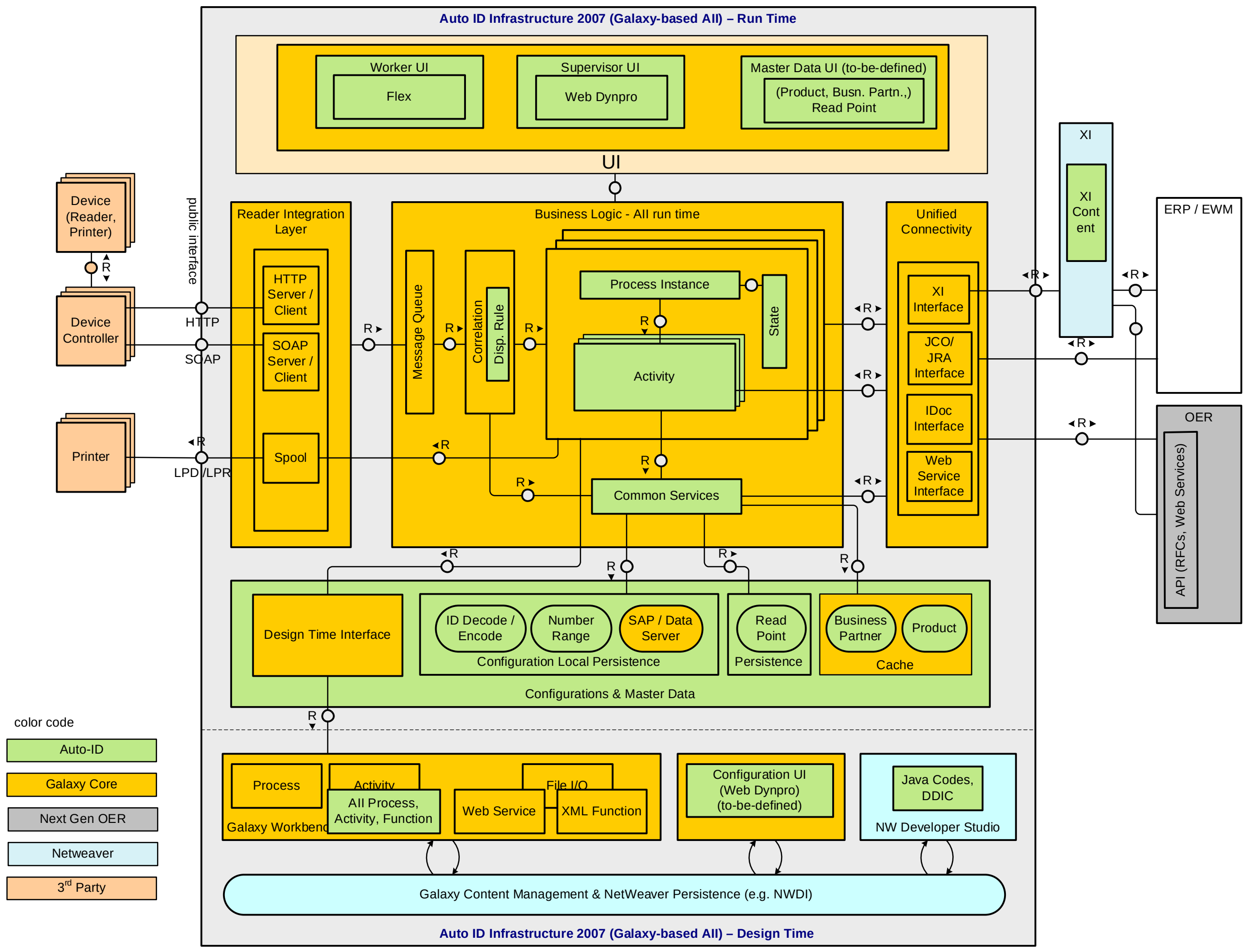


Figure 2: AII Architecture Diagram

Messages from the reader resp. device controller are received by the Web Services of the Galaxy processes. The Event Correlation of Galaxy determines based on the type and content of the message the relevant process (starting a new process instance) or already active process instance (continuing it).

The process model of them is build out of several activities or transformations using mapping functions. The combination of the activities depends on the scenario, which should be supported.

The activities call services. They are provided by the backend systems as Web Services or deployed at the Galaxy server as part of the Service Layer. Galaxy is currently restricted to the call of Web Services in the activities. For performance reasons an alternative solution e.g. EJBs will be required. The job of these last Web Services is the access to the master and transactional data and the connection to the Persistence Layer.

These activities are used to merge the information from the currently received message to the currently known business process step status (e.g. picking, loading). The status of the process step will be checked and depending on the result can the message been taken over or rejected. In both cases an appropriated feedback to a worker UI is necessary.

The status of the process instances can be monitored and influences via a supervisor UI.

NG OER Architecture

The following sequence diagram shows the data flow through the RWA platform at run-time. Emphasis in this diagram is on the NG OER. AII and external sources for the (business type) events are only shown as a single integral entity.

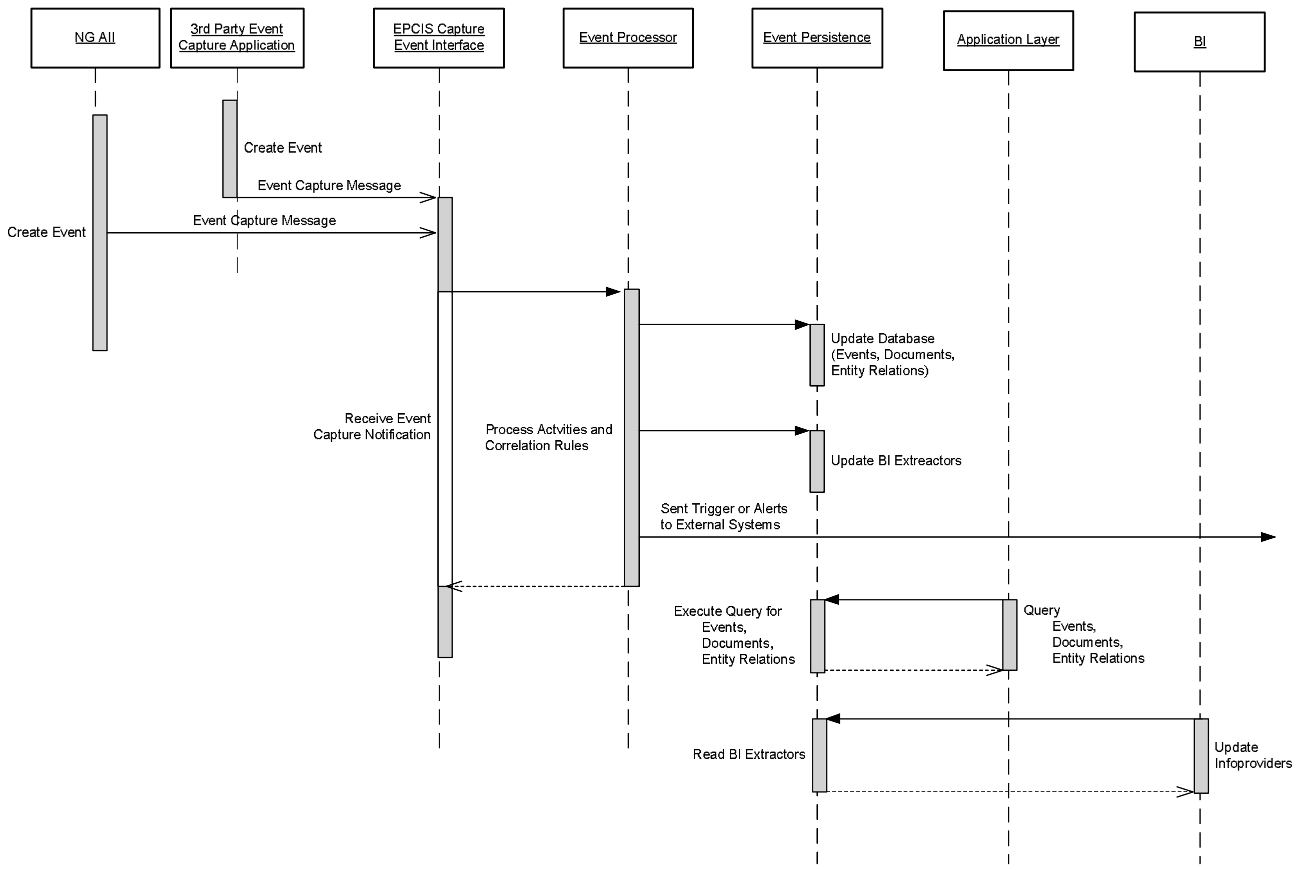

NG AII
3rd Party Event Capture Application
EPCIS Capture Event Interface
Event Processor
Event Persistence
Application Layer
BI
Create Event
Event Capture Message
Create Event
Event Capture Message
Update Database (Events, Documents, Entity Relations)
Receive Event Capture Notification
Process Actvities and Correlation Rules
Update BI Extreactors
Sent Trigger or Alerts to External Systems
Execute Query for Events, Documents, Entity Relations
Query Events, Documents, Entity Relations
Read BI Extractors
Update Infoproviders

Figure 3: OER Data Flow at Runtime

The events, independent from whether they are created by an NG AII from the SAP RWA solution or created by any external potentially non-SAP instance are received through the EPCIS compatible Capture Event Interface. Correspondingly the format of the message between NG AII or external instance and the NG OER has to be compliant or at least compatible with EPCIS. The detailed architecture will also be suitable to implement additional capture event interfaces other than EPCIS. NG AII interfaces can be used to update NG OER simultaneously. The Capture Event Interface forwards the events to the Event Processing which triggers any follow-up activity (see above for a description of the term activity). In the simplest case the only of those follow-up activities is to store the event in the Persistence Layer. As explained above "storing the event" not necessarily means "storing the event document as such". There may be some "hard-wired" logic for special event types to extract certain pieces of information and to update tables or other storage objects having a specialized structure optimized for special used cases with these pieces of information. The entity relation table is the most typical and most frequently used of these tables. The data tables to keep the information for BI extractors are also part of the Persistence Layer. Updating the BI extractor tables is not a "hard-wired" logic but effected via activity from Galaxy, because configuration is typically needed in this context. Likewise raising events or triggering task in external tools (e.g. the unified work list, UWL) is controlled via Galaxy (besides Galaxy internal task manager, of course).

In case the RWA platform is configured as an NG AII with local persistence, the business level event processing, which is typical for the OER usage type is not needed. As shown in the following diagram the Persistence Layer is then updated directly from an activity in the AII level event processing:

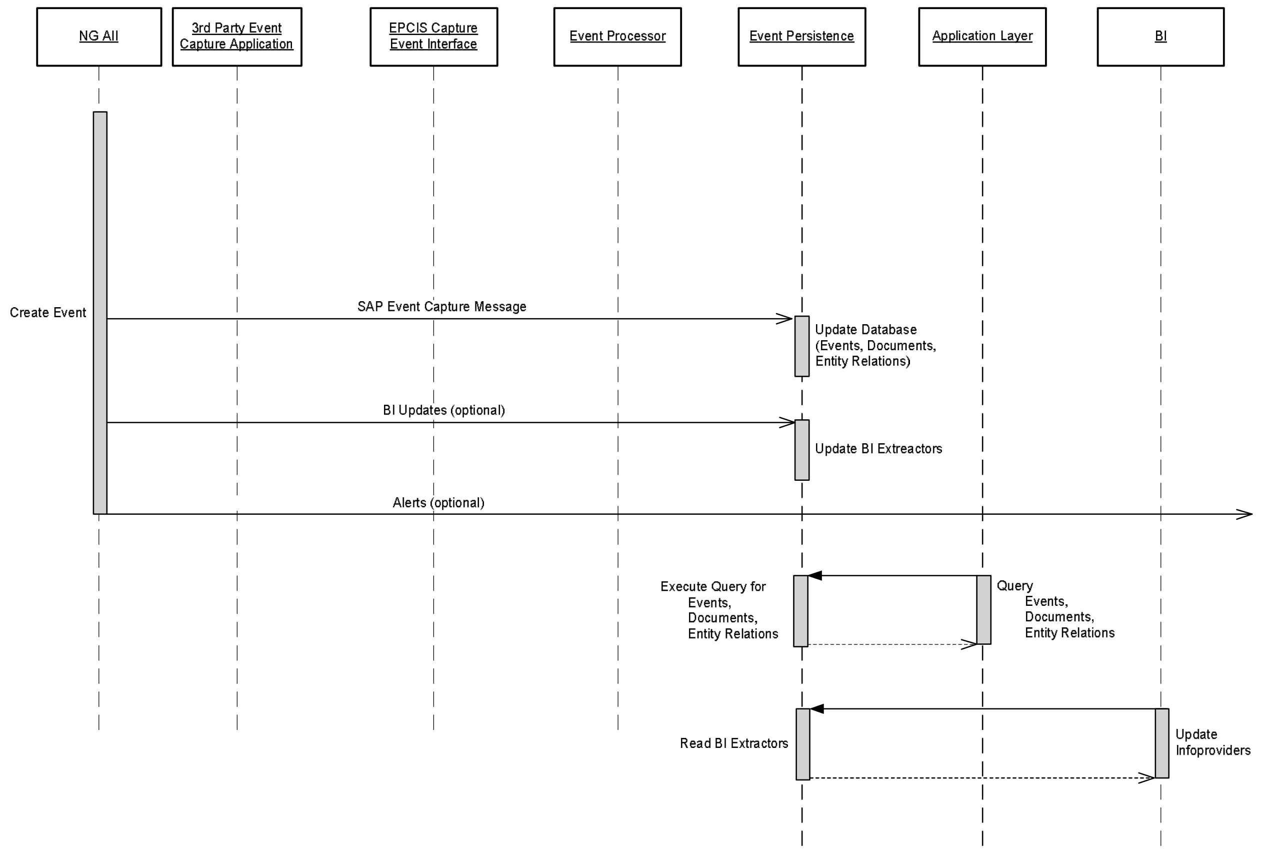

NG AII
3rd Party Event Capture Application
EPCIS Capture Event Interface
Event Processor
Event Persistence
Application Layer
BI
Create Event
SAP Event Capture Message
Update Database (Events, Documents, Entity Relations)
BI Updates (optional)
Update BI Extreactors
Alerts (optional)
Execute Query for Events, Documents, Entity Relations
Query Events, Documents, Entity Relations
Read BI Extractors
Update Infoproviders

Figure 4: Persistence Layer used by NG AII

In this case the message sent from AII to OER does not need to be compliant with EPCIS or any other industry standard. But it is important that each event of a type which shall be EPCIS compliant must be designed and stored in a way that the EPCIS defined SimpleEventQuery (i.e. one of the query services the Event Layer has to provide) can return fully EPCIS compliant information. This relates to the structure of the event as well as for the content. Likewise almost any event type defined according to any other industry standard should fit into this message channel. The attached XSD scheme shows an example for how such a generic message to exchange data between SAP NG AII and SAP NG OER could look like:
https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=8569B13642F6724F935BF6F7E3E05739.

The following block diagram illustrates a variety of typical configurations and how they communicate to each other in a fictive simplified supply chain.

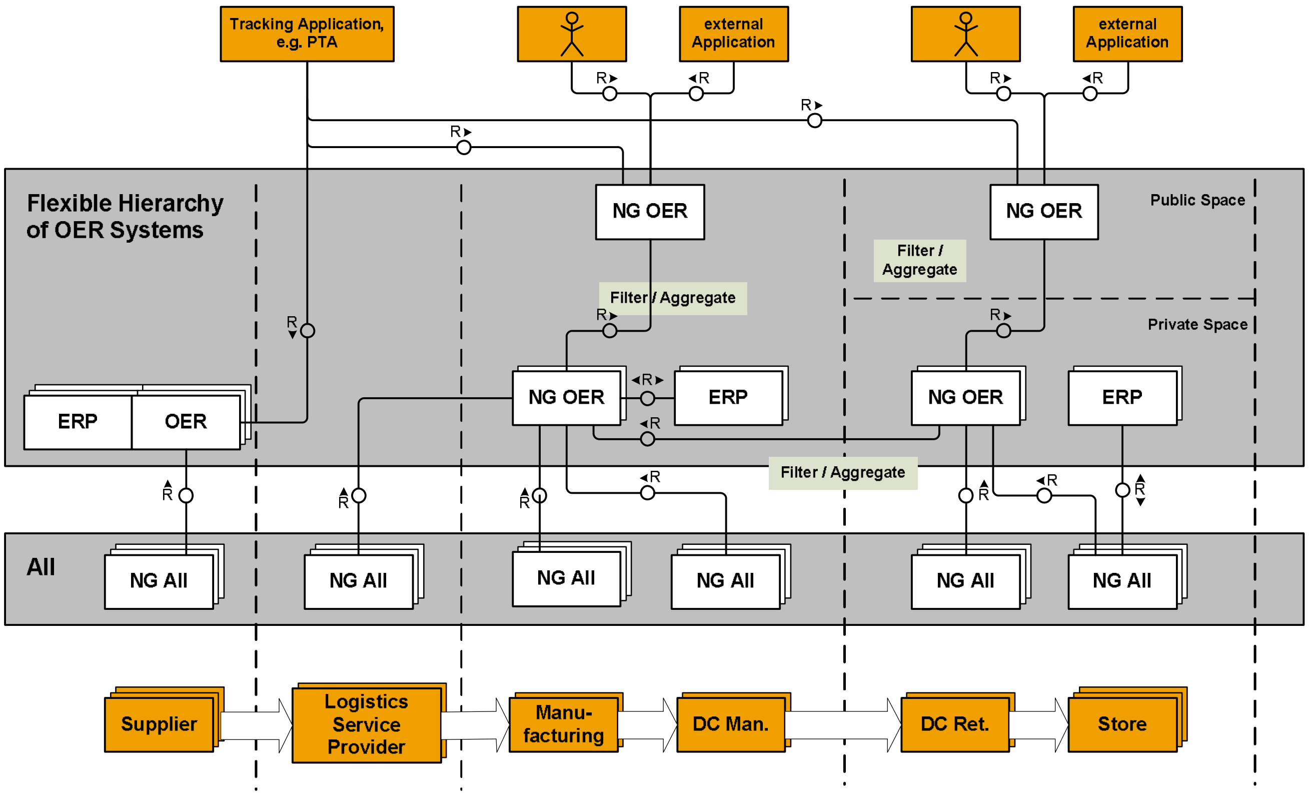


Figure 5: RWA Sample Configurations

One OER typically collects events from several connected AII systems. The AII systems may use a local instance of the Persistence Layer as a local store and replicate the events to a central OER as redundant storage, or, they may use the central OER as the only place to persist the events. The decision about one or the other option is installation specific and based on considerations such as system availability, storage and database administration costs etc. AII and/or OER may exchange data with the connected OLTP backend systems. The abbreviation ERP in figure 5 is just a placeholder of any type and any number of local backend systems (ERP, EWM, CRM, SCM, SNC, etc.).

An OER instance may be configured to forward selected events to another OER instance. This can be an OER belonging to the same organization or any OER of a different business partner and organization in the supply network. The rules and conditions to select the events that shall be forwarded to the related destination is part of the business process model in Galaxy (and hence can also be modeled graphically). It is also conceivable that information from several events is aggregated forwarded by means of a new event or document to a downstream OER. Any OER provides services to query the events and other data available therein. Applications on top of the OER may have data from a specific OER available or combine information from several OER instances belonging to several partners in the supply chain network. Tracking applications are typical examples for a cross-network application.

A typical problem of tracking scenarios, however, is that the OER instance having information about the object under investigation available is not known by the querying application when launching a query. Services such as Directory Services, ONS (Object Naming Services) or Discovery Services providing “dynamic” lookup of the destinations per tracked object would be needed in that case, but are not part of the present prototype architecture. Especially in case of the Discovery Service there is even no concept available for a potential implementation.

Interfaces

Currently the reader devices will send their messages as plain HTTP or SOAP messages. In case Galaxy is not able to receive these messages, converters have to be implemented (in Java) to convert these messages to Web Service calls. It is intended to use in future a NetWeaver adapter framework (as implemented for e.g. mobile devices or in XI) or the MDS of xMII instead.

The backend OLTP systems (in standard ERP) and the Persistence Layer are integrated at least with Web Services. In case of performance problems e.g. RFCs should be available in addition (requirement). This depends on the capabilities of the Unified Integration Layer and should be hidden in the Service Layer. At least RFCs and IDOCs are expected to be able to connect to ERP systems of older releases.

The OER will provide a number of Web Services to retrieve the data stored therein. Some of those services will be made available for access through external applications. The SimpleEventQuery service according to EPCIS is the most typically example for such a service. The definition of those services and the corresponding security concepts is not part of the actual prototype project. But the services should easily fit into the proposed architecture without principal changes.

Alternatives

- Current AII 5.1

  Instead of implementing a new product at a new platform, the existing AII 5.1 (completely implemented in ABAP) can be redesigned to reach the performance requirements.

  AII 5.1 is highly configurable (whereas some additional flexibility would be necessary), but currently no possibility exists to do this in a graphical manner (which is an additional requirement for NG AII). ABAP knowledge is necessary to understand the existing activities (by code review or debugging) or to add new one.

  A continues improvement of this implementation has the advantage, that an upgrade to the new version would be possible (depending from the type of redesign). Thru the introduction of the graphical modeling in NG AII, an upgrade (taking over the configuration or process model from current AII to NG AII) is not possible.

- Persistence Layer

  As mentioned in the chapter ‘Main Architecture Concepts and Decisions’, the distribution of the RWA platform onto two different stacks (NetWeaver 7.x ABAP and NetWeaver CE Java) is less favorable from a TCO perspective, especially because of the required administration knowledge in “both worlds”. The alternative to be evaluated in case given is to implement the Persistence Layer on JAVA, too.

Deficiencies

- Stack Switch Java ↔ ABAP

  A stack switch from Java to ABAP is necessary to request the data (master data, transactional data) from the backend systems (e.g. ERP) as well as to access the Persistence Layer.

  The access to the backend system can’t be avoided and should be ‘rare’ (only at begin and end of a process instance). The call-up points are defined by the modeled business process step itself, by business process steps executed in parallel and by the used system landscape.

  The access to the Persistence Layer is executed more often and should be as fast as possible to be able to reach the performance targets. But it is assumed, that the proportion of time used for establishing the connection is high.

  Because of the two stacks it is necessary

  - To implement, configure and maintain at least two servers (one per stack). The usage of one server for both stacks together is not recommended by SAP

  - To establish knowledge for both platforms and stacks (at least regarding developer and administrator)

- Web Service Calls

  It is not sure, if the necessary number of Web Service calls per second (estimation of about 100 to 500 calls/sec and more, depending on the used process model) can be handled by NetWeaver.

- Unique Identification

  Uniquely identifying serialized objects (or more general entities) is one of the central new aspects the RWA platform has to deal with. Unfortunately there are an arbitrary number of schemas to define unique identifiers and to encode the corresponding IDs for the labels and tags.

  Thus the RWA platform needs to provide the following functions relating to unique IDs:

  - Encoding and decoding of IDs in “technical” (binary, hex or similar) format. The same type of ID can have different encodings. A sGTIN for example can be encoded on an RFID tag by 96 or 198 bit
  - The same type of identifier can have different readable formats used for data exchange. The sGTIN, for instance, can be represented by URNs in the XML messages used for data exchange. Different URN schemes apply for this URNs, which can be used in parallel, e.g. urn:epc:id:sgtin:x.y.z, urn:epc:tag:sgtin-96:a.b.c.d, urn:epc:raw:x.y
  - The readable format is not always suitable for displaying in user interfaces. Thus a configurable, local, multi-component representation of the ID is needed
  - The same object or entity can get several IDs different regarding the underlying definition and standard assigned. The same object can, for example, get a sGTIN, an USDOD, and a 2D Barcode assigned. Typically those different IDs for the same object can’t be converted into each other by calculation. Instead often look-up tables are needed to map the IDs that point to the same object. This has performance implications because of the huge number of IDs that are generated for serialization
  - Existing system landscapes typically use internal identifiers that are only locally unambiguous. Thus at the system boundaries ID mapping are needed to convert the local IDs into unique one for cross-organizational data exchange

  Since handling of unique IDs is not specific for the RWA platform (eSOA will also need standardized unique IDs for multi-tier cross-organizational messaging) the better architecture would be to integrate the listed functions into the NetWeaver platform rather than into the RWA platform.

- Master Data

  The ERP backend system is used as a provider of material and business partner master data. A central UOM conversion service provided is currently missing. A function module has to be used instead (called via RFC).

  The only master data that has to be maintained in NG AII is related to the reader devices and their associated UIs. It is preferred, that this master data is stored and maintained in the Persistence Layer (ABAP stack), to be able to use the already existing infrastructure.

  Any master data (material master, business partner, reader) is cached in the Service Layer. The access to the different master data is provided by or via the Persistence Layer.

  Because of the restricted functionality and usage is the MDM server out of scope.

- Maintenance Java

  The maintenance of a Java application is not as comfortable as for ABAP. There is no support of notes. It seems that in case of corrections the complete archive has to be shipped again.

Debugging at customer side stops all other Java processes at this server using the same code. In a productive context this would stop all activities (using this server).

Main Architecture Concepts and Decisions (Design time)

The applications are developed based on NetWeaver CE.

- The NetWeaver Development Infrastructure (NWDI) with the NWDS is used to design the Galaxy process models and to implement the activities and functions in Java needed in these models. Also the Java part of the Service Layer is implemented therewith
- The ABAP Workbench is used to implement the Persistence Layer
- Web Dynpro Java is used to implement the UI of NG AII and NG OER
- Flex is used to implement in the case of NG AII worker UIs

  Flex is used here, because the requirements (large buttons, touch screen support) at these UIs can't be fulfilled by Web Dynpro.

Total Cost of Ownership

The benefit of RWA (regarding usage types NG AII and NG OER) for customer increases with integration to backend systems like ERP. In current AII architecture such integration is designed mainly for communication via XI. For customers with complex system landscapes XI makes utmost sense. For customers with e.g. only one ERP system who wants to leverage NG AII capabilities for RFID enablement, XI is just not necessary. For this reason we are evaluating new communication scenarios during prototyping, scenarios with direct communication using standard ERP interfaces and Web Services.

Another topic of TCO reduction is reached by providing a lightweight product for installation in many local locations. Lightweight means that RWA product needs to run on a PC. For this purpose we are using Galaxy which is available on lean NetWeaver CE stack.

Furthermore Galaxy is used to reduce cost of modeling new processes, by leveraging the flexibility of these tools to model new processes in short time.

Deployment

The delivery to the customer consists out of

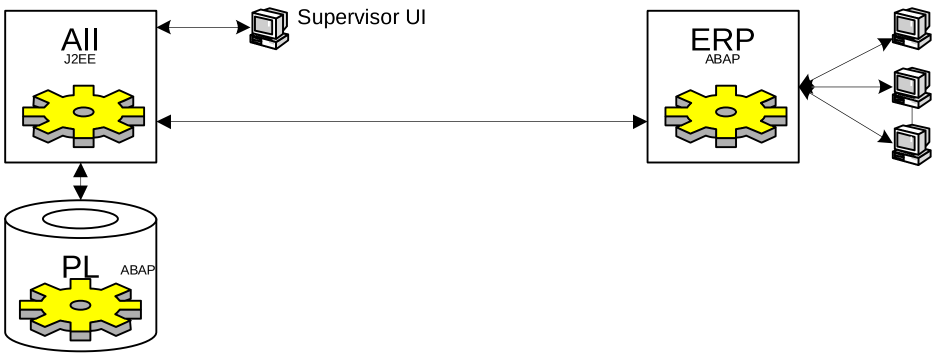
- Java Archive (EAR) containing the
  - Service Layer
  - Implementation of the activities and functions used (or usable) in Galaxy models
  - Galaxy Process models
- ABAP transport of the Persistence Layer

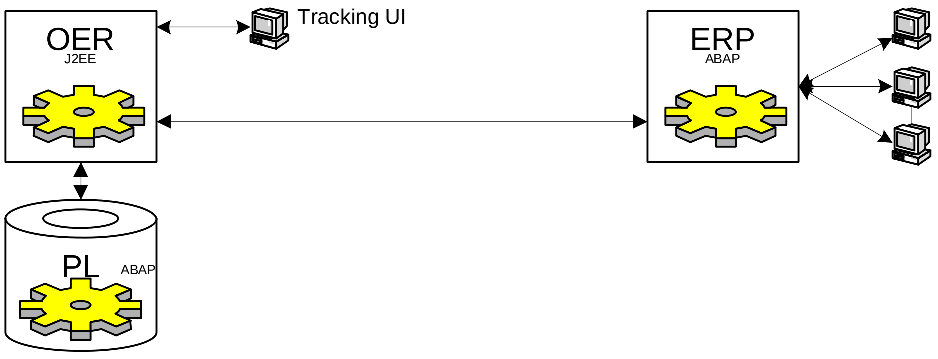
Both have to be installed together, with the correct version because they are relying on each other. In a multi system landscape as intended for the complex deployment (slide below), this can lead to sever integration problems. The central ABAP persistence has to work with different releases of Java runtime at the same time.

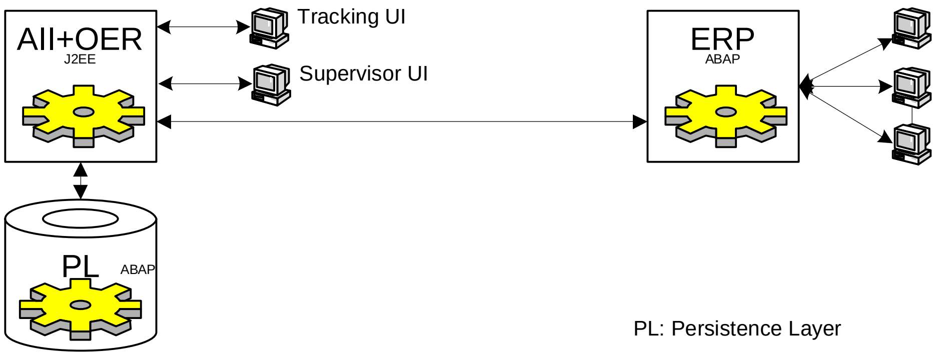
Following deployment scenarios are supported:

- Minimal Deployment

  One RWA platform will be connected to one ERP system. The RWA platform can be driven with one or both usage types at the same time.







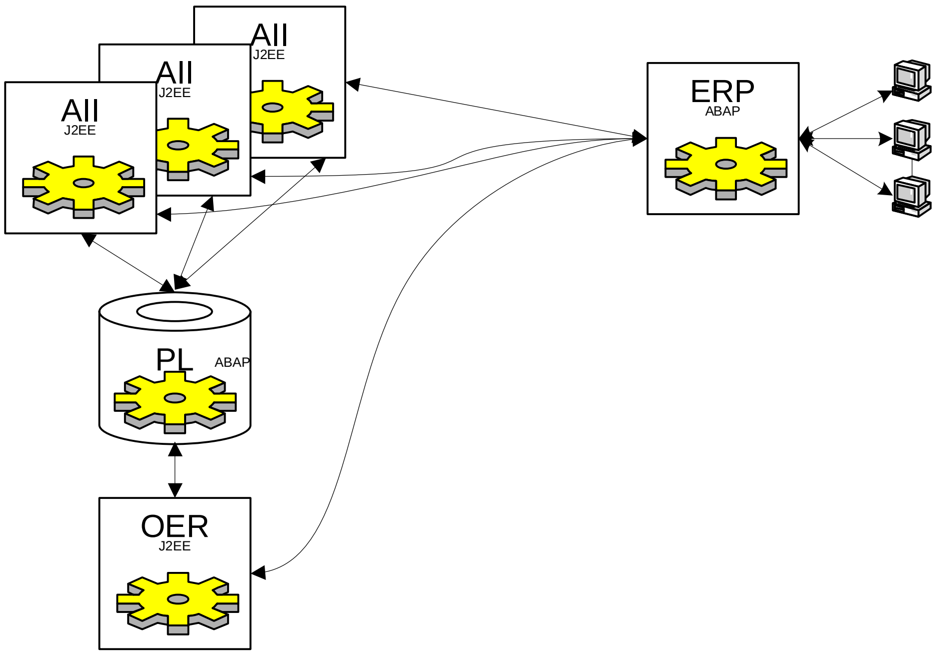
PL: Persistence Layer

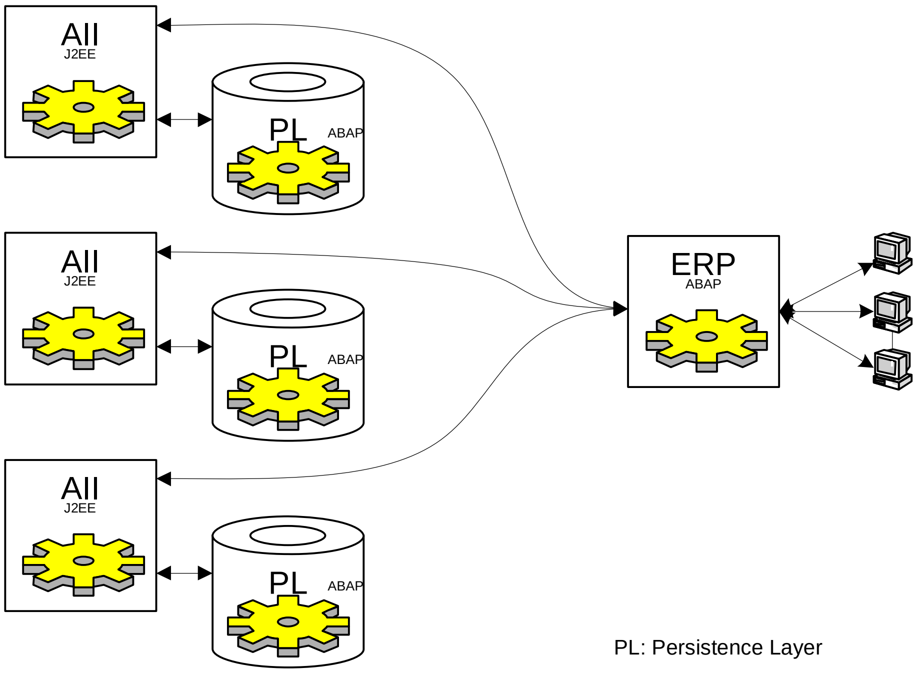
The reader devices (necessary for NG AII) and their UIs or the sources of the events (NG OER) are not displayed.

In rare cases also several ERP systems can be connected to one RWA platform.

- Complex Deployment

  Out of performance reasons it has to be possible to install RWA also at several (independent) systems. All this independent systems are connected to the same ERP system. The AII systems can use together a central persistence or each a local one.





PL: Persistence Layer

As mentioned before, any of the RWA platform systems can be used with one or both usage types (to simplify the diagrams isn’t any combination and none of the UIs shown).

Configuration and runtime split

If in a system landscape several RWA platform systems are driven with the same usage type (e.g. three NG AII systems), one of these systems can be used as configuration master system.

This master system is used to design, configure and test/simulate the processes (in Galaxy). The configuration and implementation (e.g. customer specific services, implemented in Java) will be transported to the other systems in the system landscape.

A subsequent adaptation of the setup can be necessary in the target systems (e.g. adaptation to the location related data).

Also in this system landscape a mixture of the different releases has to be possible. The RWA platform and Galaxy have to be able to take over configuration from older releases.

This distribution pertain only the configuration and not the installation of an application. In case of configurations including a Persistence Layer the distribution of configurations may have some restrictions, because the automatic distribution and remote installation of a database with preconfigured settings is not foreseen.

Maintenance

In general the NG maintenance strategy will be like for all SAP standard products, but finally to be defined for the product after successful prototyping and clarification of final architecture. In case of objects designed within the ABAP Workbench is the strategy defined, for all other objects (Java code, Galaxy process models) it is open.

Based on current architecture of NG prototype the maintenance efforts could be quite high: Knowledge is necessary at following areas:

- NWDI including NWDS
  - Galaxy
  - Java
  - Web Dynpro Java
  - XML
- Adobe Flex (worker UI of NG AII)
- ABAP Workbench
  - ABAP
  - DDIC
  - XML database

Known disadvantages in maintenance of Java applications:

- No concept like SPAU available
- Debugging and error solving time consuming and may lead to downtimes on customer site

Architecture Documentation

Product Definition

This document https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=DEF4D7CA21F66B4CAC361C59CBFCE27D describes the NextGen scope of the SAP Auto-ID Infrastructure as it may be realized in a number of phases. The key proposition is to split the SAP Auto-ID Infrastructure in a design-time and runtime framework.

Specifications

NG AII and NG OER Use Cases: This SRS
https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=80A3579E63490D4EAC0D6D75576B2787 has been derived as a set of initial NG AII and NG OER prototype deliverables to ultimately provide RWA platform capabilities that are in line with the approved three phased Real World Awareness Project Charter and associated 3-year investment bucket.

The goal of the NG AII and NG OER efforts is to deliver a cohesive Real World Awareness Platform that may serve SCM, ERP and in the future perhaps AP. The prototype efforts focus on technical and performance evaluation based on basic use cases described in the specification.

NG AII Prototype Use Cases:
https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=6A7B74F9F23BC0409D34D16DFB0386B8

The scope of this SRS
https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=A306BF0EED25304290D2993F5B8F4EA0 is to cover NG AII prototyping, not NG AII. NG AII prototype contains the following building blocks:

- NG AII design time – RFID process “modeling”.
- RFID specific action blocks to support NG AII use cases
- Event persistency layer interface
- Master Data
- ERP integration
- Test tool
- Performance
- Services for connection the ‘unload UI prototype’ with AII backend

SRS for the NextGen OER Prototype:
https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=B7D6801C982CCE48A61F6A699AD4B6BB

Designs

Until now (status November 2007) no design documents are written for NG AII. For NG OER an initial design document has been created that will be updated step by step whenever a design decision is made.

III. OPEN ISSUES, OUTLOOK AND RISKS

Open Issues and Outlook

Open Issues

- The XML features provided by IBM DB2, Oracle and Microsoft SQL are not supported by the Open SQL layer in NetWeaver and at present not available in MaxDB. Thus for the product development special approval is needed regarding SAP product standards and implementation of some DB specific services.
- Performance

  The central objective of the prototype project is to evaluate whether the high performance KPIs especially regarding OER (see SRS NextGen OER) can be fulfilled or not. Regarding OER performance there are two major concerns:

  - Performance of execution of the modelled processes in Galaxy especially with regard to the service calls to update the event persistence and the service calls to retrieve historical data from the event persistence during processing
  - Performance of the XML tables and XML features in the named databases especially with regard to indexing, backup and archiving
- Performance of stack switch

  The influence of the stack switch (from Java to ABAP and backward) to the performance can't be assessed in detail until now.
- Performance of Web Services

  The maximal reachable number of Web Service calls per second is currently unknown: The latest known number is 30 calls/sec to the ABAP stack, but it couldn't be verified until now. Also the context of this measurement is unknown. It is also not known, if this can (and will) be improved by NetWeaver.

  In first estimations about 100 to 500 calls/sec and more, depending on the used process model are necessary to reach our performance goals.
- Definition of 'lean'

  A final definition of 'lean' is missing.

  - o Is the application lean enough, if two systems have to be installed to use this application?
  - o Is NetWeaver CE lean? Will NetWeaver CE stay lean in the future?

  The TCO has to be as small as possible. The option to deploy the runtime on a reader or an embedded device currently not exists, because NetWeaver CE is still too huge. A basis like OSGi or Mobile Infrastructure would be required.
- Where to place the configuration (e.g. Read point definition; assignment of logical system name to system address)?

  Until now it is not decided, where the configuration will be stored:

  - ABAP stack (as part of the Persistence Layer):

    + The generation tool can be reused (to get a standard maintenance dialog)

    + The existing logging tools, transport landscape and maintenance tools can be used

    + Business Configuration (BC) Sets can be used to document the configuration

  + Archiving is available
  - Master Data has to be cached at the Java stack (out of performance reasons)

- Java stack (e.g. as part of the Common Service Layer):

  The advantages from the usage of the ABAP stack are the disadvantages of the Java stack and vice versa.

  + Configuration and application are at the same stack
  - No transport of content possible (as standard solution)

- Unique ID Management and Service Engine (scheme management and encoding and decoding as well as mapping services)

  A SAP wide central Unique ID Management is missing and has to be implemented.

  The Unique ID Management has to be callable from Java and ABAP as well.

Prerequisites from other parties

- Galaxy

  The fulfilment of the requirements of NG AII and NG OER depends from the timeline and the completeness of the implementation of the RWA requirements to Galaxy.

  The currently placed requirements are listed in https://ifp.wdf.sap.corp/sap/bc/bsp/sap/edms_files/edmsviewfilesession.htm?filename=dummy.pdf&refId=55C2E3591B80244885A81621A8305280&type=PPO&relId=L&objId=9058DA24BA2820409EC934850FAB3518. The timeline is unsafe.

- Unified Integration Layer

  The exact scope and the availability of the Unified Integration Layer are currently unknown.

  It is expected, that at least Web Service, RFCs and IDOCs are supported. It should be possible to integrate the adapter framework into it.

Alignment

Currently is only an alignment with Galaxy/NetWeaver CE necessary.

Planned Next (status November 2007)

A prototype will be implemented until end of April 2008 to prove, if the planned performance goal can be reached. The prototype will be based on current version of Galaxy. The activities and functions will be implemented in Java. The Persistence Layer will be implemented in ABAP and will use the XML DB features from IBM DB2.

Outlook

- **Modelling**

  The modeling of the processes in Galaxy should be embedded into a multi layered interconnected end-to-end modeling (ASM wording) using MOIN.

  - Location modeling (plant, enterprise structures)
  - Definition of readers and read point
  - Floor plan based location modeling with assignment of reader locations and event type and inferred subsequent business context
  - Business Process modeling and templates

  - R/W data model per RFID I-Point

- **Occasionally connected System Landscape**

  Allow NG AII also to work in occasionally connected system landscapes (compare to mobile devices), i.e. the backend system can't be reached anytime. A buffer structure has to be provided to cache a workload list (list of announced deliveries, which are expected to be handled in the next time).

- **Integration of the Generic Device/ Sensor Connectivity Layer**

Risks

- Modelling of Exception Handling in Galaxy

  According to the current state exception handling is part of the model that is defined in Galaxy. This means business modelling and exception handling are part of the same model. This quickly results in complex model that are difficult to read and edit.

- Project Schedule
  - Solution management expects to have RWA platform prototype finished in 2009 and the product at market in 2010. Since Galaxy is a central building block of the architecture the project schedules for both the prototyping and the final product strongly rely on the availability and functional completeness of this tool. At present there is no clear commitment for several of the central requirements.
- Technology
  - Missing developer skills: lack of experience in developing full fledged Java EE applications
  - We face all risk related to SAP wide known lack of functionality of Java design and runtime environment
    - Downtimes for debugging and error solving
    - Restricted toolset for maintenance
  - Brand new Galaxy and NetWeaver CE
  - Manifold technologies used (Galaxy, Java, ABAP, etc) lead to a quite complex architecture, and especially place a challenge while meeting the hard performance and TCO requirements

ABBREVIATIONS

AII	Auto-ID Infrastructure
BI	SAP NetWeaver Business Information Warehouse
EAR	Enterprise Application Archive
EJB	Enterprise Java Bean
EPCIS	Electronic Product Code Information Services
ERP	Enterprise Resource Planning
MDS	Manufacturing Data Services (xMII)
NG, NextGen	Next Generation
NWDI	NetWeaver Infrastructure
NWDS	NetWeaver Developer Studio
OER	Object Event Repository
OLTP	Online Transaction Processing system
OSGi	Open Services Gateway initiative
RFID	Radio Frequency Identification
RWA	Real World Awareness
TCO	Total Cost of Ownership
UWL	Unified Work List
xMII	xApp Manufacturing Integration and Intelligence
XML	Extensible Markup Language