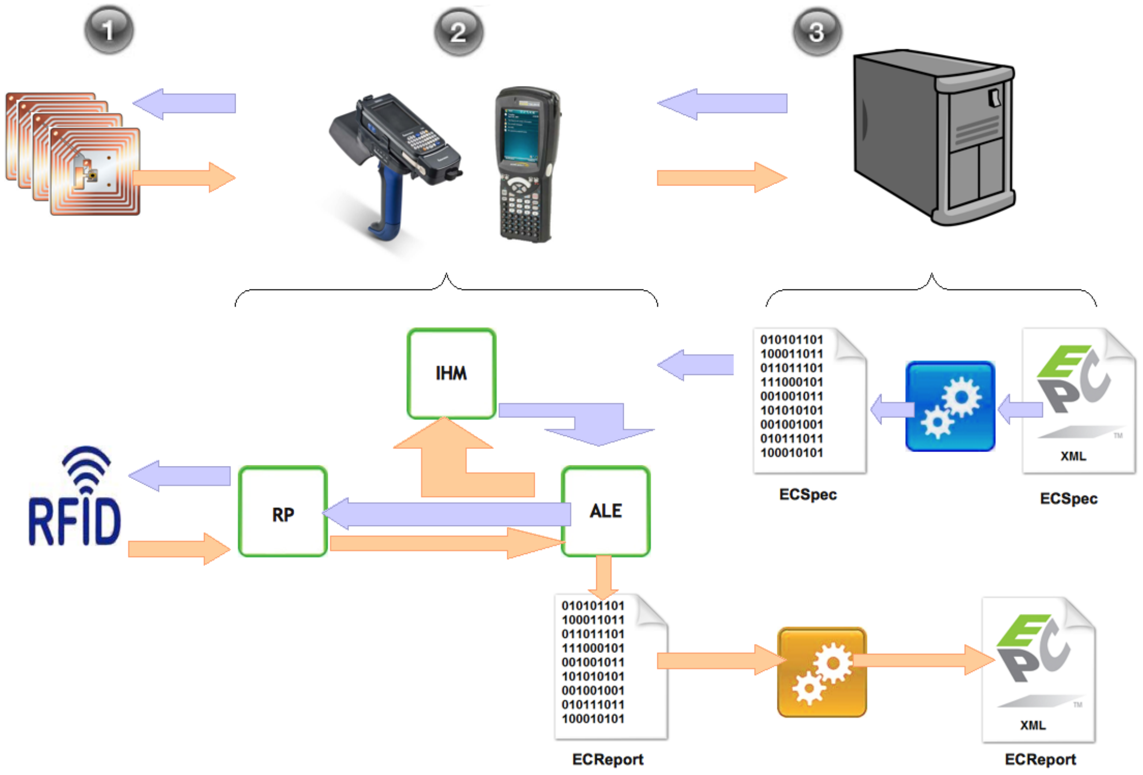
Embeded ALE
Embedded ALE

This section describes the Embedded application software developed by the POPS team, Inria-Lille. This software includes a light-ALE which can be implemented in mobile terminals such like PDA (we choose the brand Psion and Intermec in our tests).

Application Context

The use case that drives the application context can be summarized as follows :



There are three main levels in this scenario :

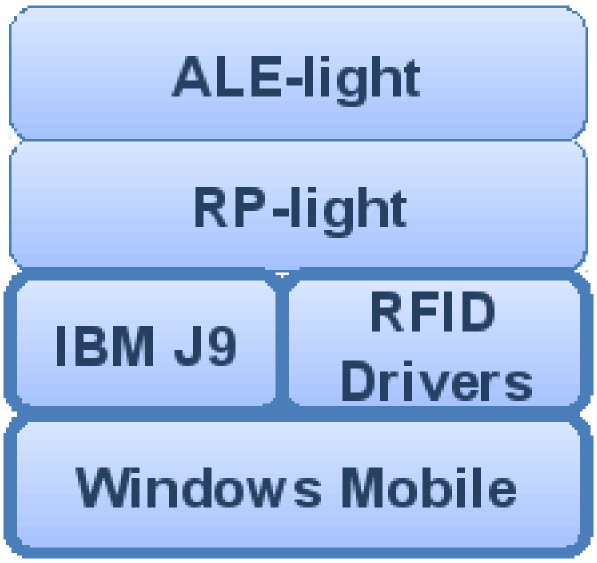
1) Tags : EPC C1G2 tags sticked on items or location spots.

2) Embedded applications :

- IHM : the user's interface for inventory management : start, pause, stop, download ECSpec and upload ECReports in a binary serialized fashion. This application is used as client for the ALE (not covered by this document
- ALE : The filtering and collection server. Tailored for use in an embedded Java environment.
- RP : the reader abstraction layer using the EPC Reader Protocol API. It interacts with the RFID reader device.

3) Server : connected to an EPC network, it hosts specs and manages reports. The XML marshalling of ECSpec and ECReports objects is performed on the server for performance issues (not covered by this document).

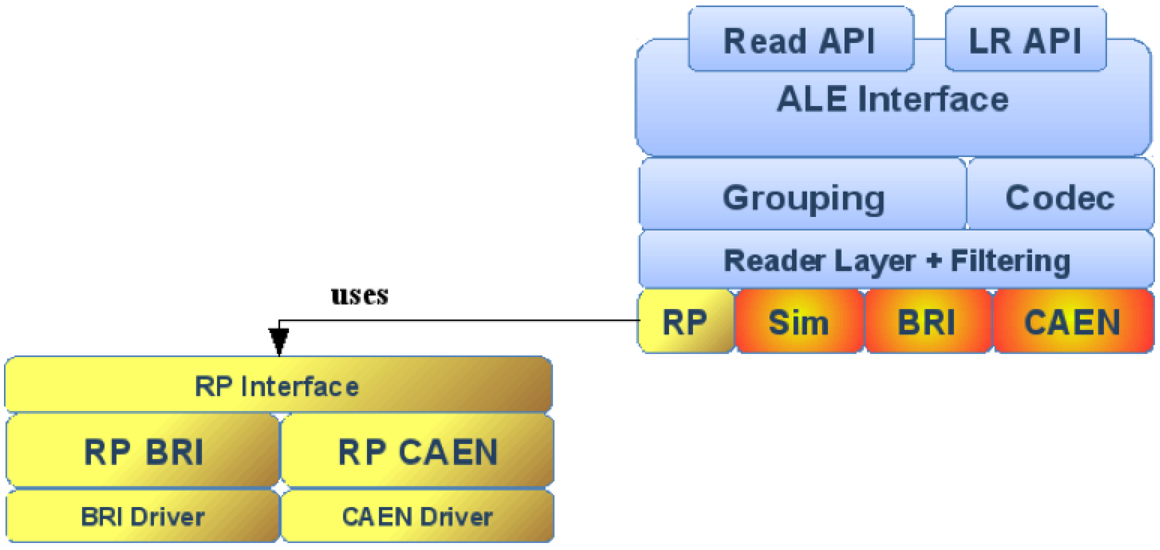
The embedded applications are fully written in Java, under the J2ME CDC 1.0 profile. They run on top of IBM's J9 virtual machine.



The use of the Java language constrains the development style in order to cope with garbage collection (such as reusable objects). Finally, the RFID readers (CAEN and BRI) are accessed using vendor-specific drivers.

Embeded ALE overview

The currently implemented layers are the following :



1) ALE Layer : Inventory of tags in reader's field, reader configuration.

In order to provide minimal inventory services, at interface level, subsets of the Reading and the Logical Reader APIs are implemented :

- Immediate mode : sufficient for user-triggered inventory.
- Fixed readers configuration : only some properties (Power, Session and InitialQ) can be updated.

The ALE engine manages tag grouping and filtering according to EPC standard patterns in input ECSpec objects. A lightweight custom CODEC was developed as well, in order to decode tag IDs using binary format (array of bytes) and in a garbage-free fashion. A filter engine is also made available for software filtering of tags. This leaves the choice for Reader Connectors to choose the best tradeoff between software and hardware filtering. Note that, because of the Java CDC constraint, the ECSpec and ECReports classes and subclasses were written manually despite of automatic generation from XSD files.

2) Reader Layer : Abstraction layer of reader device.

This abstraction layer defines the contract that a reader connector shall respect in order to be bound to the ALE. It provides several services to the connectors such as access to the software filtering engine. Four connectors has been developed :

- RP : connector to an RP-compliant reader implementing the minimal inventory functions.
- Sim : a custom reader for debugging purposes. The list of tags for simulation can be retrieved from a configuration file generated by the tag generator application (see next section).
- BRI : Connector to the Intermec IP30 reader using the BRI driver.
- CAEN : Connector to the CAEN A528 reader using the CAEN driver.

3) EPC Reader Protocol wrapper (RP) : This wrapper defines the Reader Protocol interface classes that are used to dialog with an RP-compliant reader device. Based on each vendor-specific driver, two implementations were developed in order to provide minimal required services (inventory). Note that the communication with the Reader Device is done locally and directly via method calls. This avoids overhead when using MTB layers for message bindings.

Users Guide

DOWNLOAD & RUN You can download the application files from here. All the source can be found here.

- Implementation in a PDA :
    - Copy the repository located in *Light_ALE_1.0* (depends on the testing PDA) to the root path of PDA.
    - Configure the file *testConfig.txt*
    - Click LaunchDemo.lnk to test light ALE (light RP included) or TestRp.lnk (test only the light RP)
- Simulation test
    - Set *simu=true* in testConfig.txt
- All the sources are zipped in *workspaceICOM.zip*

Certain complex classes were tested unitary by simply adding a main method in the class that tests the main functions (ex. pops.ale.engine.collect.grouping). This is useful for unitary tests on PDA.

The RP-light subset was tested on the PSION and INTERMEC PDA's using the test.Test class.

The ALE-light tests contains more information useful for the ALE user. The tests are written in the **pops.ale.engine.TestCase class**. The test Case is configured in an input file **(Config.txt)**, in order to choose the test, select the reader layer...etc

There are 5 tests (Test 3 is the whole ALE test) available :

- Test 1 : Validates the CAEN API in order to read on the PSION.
- Test 2 : Tests the grouping algorithm used in the ALE engine.
- Test 3 : Tests and demonstrates the immediate mode (spec build, call, use report).
- Test 4 : Benchmarks for the beep durations (Searching algorithm which is used to look for an item in the neighborhood by the beep frequency of the PDA)
- Test 5 : Tag Searcher (Geiger-like) proof of concept (Geiger.java class).

More details about the embedded light RP can be found here.

Last Author: xwiki:XWiki.reckeyzhang Date: 2010/06/14 14:20
Copyright (c) 2008-2010, Aspire