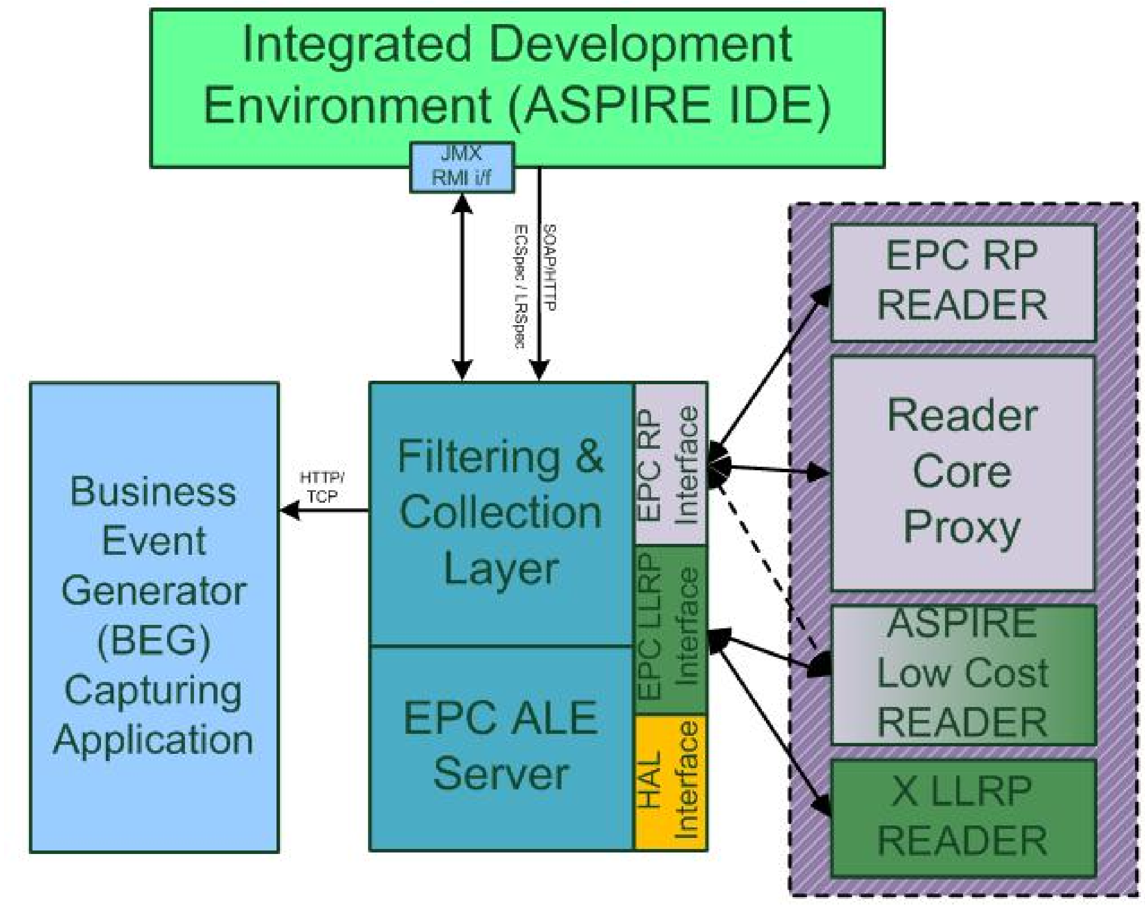
Aspire :: Filtering & Collection
Filtering And Collection Server

- The AspireRfid Filtering and Collection Middleware is a slightly modified version of Accada RFID Middleware, which is now called FossTrak, licensed under LGPL. This component implements EPCglobal's ALE 1.1 specification. AspireRfid have implemented changes and bug fixes on the original Accada middleware. As of 14/05/2009 we have ceased distribution of FossTrak licensed code.
- Deployment of Filtering And Collection Server on OSGi
- CiliaALE : a lightweight, flexible and dynamically reconfigurable ALE Servers for Android phones and tablets (developed by the Adele team, LIG, Grenoble)
- Embedded ALE: developed by the FUN team, Inria-Lille, which can be implemented in mobile terminals.
- Extended TDT ALE: developed by the FUN team, Inria-Lille, which is compatible with different input identification formats (i.e. EPC, GS1, ISO, MAC address, Telephone numbers, etc.).
- Light ALECC: developed by the FUN team, Inria-Lille, which aims to supply an easy and efficient framework for developers to realize the most used basic operations defined by the EPC ALECC (writing API) standard.
- LLRP toolkit library (updated according to LLRP1.1): developed by the FUN team, Inria-Lille, which aims to provide an updated LLRP toolkit library for developers intersted by LLRP interface.

AspireRfid Filtering And Collection Server

Introduction



In the scope of large scale deployments, RFID systems generate an enormous number of object reads. Many of those reads represent non-actionable "noise." To balance the cost and performance of this with the need for clear accountability and interoperability of the various parts, the design of the ASPIRE middleware seeks to:

- Drive as much filtering and counting of reads as low in the architecture as possible.
- Minimize the amount of "business logic" embedded in the Tags.

The Filtering and Collection Middleware is intended to facilitate these objectives by providing a flexible interface (ALE (Application Level Events) interface) to a standard set of accumulation, filtering, and counting operations that produce "reports" in response to client "requests." The client will be responsible for interpreting and acting on the meaning of the report. Depending on the target deployment the client of the ALE interface may be a traditional "enterprise application," or it may be new software designed expressly to carry out an RFID-enabled business process. but which operates at a higher level than the "middleware" that implements the ALE interface. In the scope of the ASPIRE project, the Business Event Generation (BEG) middleware would naturally, consume the results of ALE filtering. However, there might be deployment scenarios where clients will interface directly to the ALE filtered streams of RFID data.

The AspireRfid Filtering and Collection Middleware is a modified version of Accada RFID Middleware, which is now called FossTrak, licensed under LGPL. AspireRfid have implemented changes and bug fixes on the original Accada middleware.

The actual AspireRFID ALE version is taken from Fosstrak filtering and collection module's revision number 2295 (released on February 2013). It implements EPCglobal's ALE 1.1 and LLRP 1.1 specifications. If we consider a user point of view, ALE compromises 3 modules as follows:

- the filtering and collection server
- a standalone clientto configure filtering and collection servers

- aweb-based clientto configure filtering and collection servers

However, if we consider a developer point of view and focus on the source code, ALE is also organized in 3 modules :

- fc-client: Filtering and Collection Client: a Java Swing based test client to execute Application Level Event specification (ALE) commands on a reader or component that implements the ALE specification
- fc-commons: common functionality used by the different Filtering and Collection modules
- fc-server: an implementation of EPCglobal Application Level Event (ALE) specification. An LLRP interface insures communication procedure with LLRP compliant RFID readers. For readers that do not support LLRP, ALE uses the Hardware Abstraction Layer (HAL) interface.

Users Guide

Requirements

Hardware (minimum)

- P IV 1.2GHz or equivalent
- 512 MB Ram
- 50 MB free HD space

Software

- LLRP Commander
- Rifidi Reader Emulator
- Apache Tomcat Java Servlet Container

Files

- a copy of the fc-server-1.0.0.war or higher (note: LLRP support requires >= 1.0.0)
- a copy of the fc-webclient fc-webclient-0.4.0.war or higher
- a copy of the aspireTcpMessageCapturer
- a copy of this LLRP ADD_ROSPEC message (to be saved in xml format)
- a copy of this ALE ECSpec message (to be saved in xml format)
- a copy of this ALE logical reader definition file (to be saved in LLRP format)

Deployment

Tomcat Deployment Tomcat is used in order to deploy the Filtering and Collection server. Deployment is worked out as follows.

- Install Apache Tomcat if you have not installed it yet.
- Copy the two WAR files (fc-server-VERSION .war and fc-webclient-VERSION .war) into the Tomcat's webapps directory and start Tomcat. The war-file will be deployed into a new folder.

For Windows users : You will usually find the webapps folder inside the tomcat installation directory :

- c:/Program Files/Apache Tomcat/webapps for 32-bit Windows
- c:/Programs/Apache Software Foundation/Tomcat/webappps for 64-bit Windows

For Linux/unix users : You can use this command to start tomcat : service tomcat7 start

The webapps folder will depend on your distribution, here are some possible locations:

- /var/lib/tomcat/webapps
- /usr/local/lib/tomcat/webapps

Once the two war files are in the webapps directory, Tomcat will automatically deploy them. The ALE server is now ready to be configured at your needs.

Rifidi configuration

- Install the Rifidi emulator
- Start the Rifidi emulator
- Instantiate a new reader of type "LLRPReader" that listens on port 5084 (default) using the Reader Wizard
- Start the reader just instantiated (right click "Start reader")

We will return to the LLRP reader emulator once the other components are configured.

Configure the Rifidi Emulator by using LLRP Commander

- LLRP Commander is an Eclipse plugin. To install it, you can "check for updates" via the Eclipse or "install a new software"

Eclipse Update Site URL: http://fosstrak.googlecode.com/svn/llrp/eclipse_update_site

- Start Eclipse and customize the view to LLRP Commander (In Eclipse: Window/Open perspective/Other/LLRP Commander).
- Create a new reader (Under Reader Explorer tab : Right Click on DEFAULT and Add RFID Reader).
- Connect to the remote adapter instance running on the ALE. Check, if you can send and retrieve messages from the reader by sending a GET_READER_CAPABILITIES message (Right click on the reader and Send message). You can follow the reader behavior towards the messages in Rifidi console and also in the "LLRP Messagebox" in Eclipse.
- Create a new LLRP message (Right click on LLRP_CMDR/New/Other/LLRP/LLRP Message)
- Edit the content (in XML Editor) by the content of the fileRO_SPEC and save
- Send the message to the Rifid Emulator(use the button "Sending the current message")
- From the Reader Explorer tab (right-click on the reader name), select "Send ENABLE_RO_SPEC message" to instruct the reader to enable the ROspec just loaded.

Now we are sure that the virtual reader is configured, we just need to provide it with tags. This step will be done after.

GUI Setup : TCP Message Capturer

This interface will spy the reports sent by the filtering and collection server, it displays incoming HTTP requests. Make sure the provided port (9999) is not used by some other application.

- Uncompress the aspireTcpMessageCapturer archive
- Start the TCP Message Capturer that listens on port 9999

Command to Start aspireTcpMessageCapturer :

```
java -jar aspireTcpMessageCapturer-VERSION -jar-with-dependencies.jar #For Example java -jar
aspireTcpMessageCapturer-V0.1m-jar-with-dependencies.jar
```

ALE web-client configuration

The web-based client is used to configure the filtering and collection server. We need first to tell it the URL through which the filtering and collection server can be accessed. Start your web browser and point it to the address of the web-based client.

```
http: //<SERVER>:<PORT>/<WEBCLIENT_VERSION>/services/ALEWebClient.jsp#For Example http://
localhost:8080/fc-webclient-1.2.0/services/ALEWebClient.jsp
```

In the next step, specify two endpoints that tell the web-based client where the "Filtering and Collection API "and the "Logical Reader API" can be found. Set the endpoint to the filtering and collection server by selecting the "setEndpoint(String endPointName)" method in the "Filtering and Collection API". Click "Invoke" to execute the command.

```
endpoint: http: //<SERVER>:<PORT>/<FCSERVER_VERSION>/services/ALEService#For Example http://
localhost:8080/fc-server-1.2.0/services/ALEService
```

To be sure that a connection between the web-based client and the server can be established, click "getVendorVersion()". A version number should be returned.

Set the endpoint to the filtering and collection server's Logical Reader API by selecting "setEndPoint(String endPointName)" in the "LogicalReader API". Click "Invoke" to execute the command.

```
endpoint: http: //<SERVER>:<PORT>/<FCSERVER_VERSION>/services/ALELRService#For Example http:
//localhost:8080/fc-server-1.2.0/services/ALELRService
```

Click "getVendorVersion()" to be sure that a connection between the web-based client and the server can be established, a version number should be returned.

Set the readers connected to ALE

The next step is to configure the fc-server with the LLRP reader connected. Click on "define(String readerName, LRSpec spec)" in the section "LogicalReader API". DO NOT CONFUSE the define method for an EventCycle with the define method for a logical reader! For this tutorial name your reader "LogicalReader1" and use theLLRPReader.xmlas your LRSpec.

```
readerName: READERNAME specFilePath: PATH_TO_SPEC/SPEC_NAME.xml #For Example readerName:
LogicalReader1 specFilePath: /home/epc/LLRPReader.xml
```

Verify that the reader has been created by clicking on "getLogicalReaderNames()". The call should return a list of logical readers in brackets. Make sure that the reader just creates is listed. In the LLRP reader emulator console you can now see the message exchange with the Fosstrak ALE middleware. You can inspect the logical reader specification LRSpec by the method "getLRSpec(String readerName)".

Define the Filtering and Collection Behavior

The behavior of ALE is described in an xml file. So in this step you will define an ALE ECSpec, this tells the ALE Middleware how the RFID tag reads ariving from the reader should be filtered and aggregated, for example: added, current or deleted tags. Click on "define(String specName, String specFilePath)" in the section "Filtering and Collection API". DO NOT CONFUSE the define method for an EventCycle with the define method for a logical reader! For this tutorial name your EventCycle "specCURRENT" and use thisECSpec_current.xmlas your ECSpec.

```
readerName: SPEC_NAME specFilePath: PATH_TO_SPEC/SPEC_NAME.xml #For Example readerName:
specCURRENT specFilePath: /home/epc/ECSpec_current.xml
```

Verify the correct defintion of your ECSpec by invoking the method "getECSpecNames()". You should get a list of ECSpec names currently defined in brackets. Make sure that the ECSpec you just defined is listed. You can inspect the event cycle specification ECSpec_current by the method "getECSpec(String specName)".

Specify the Event Consumer of the ALE Events Being Sent

When there is no subscriber for an ECSpec, the ECSpec is not executed. We therefore need to specify a listener by subscribing our event sink to the ECSpec "specCURRENT" we added earlier. Invoke "subscribe(String specName, String notificationUri)" and register the URL on which the event sink GUI is listening.

```
notificationURI: http: //SERVER:PORT specName: SPEC_NAME #For Example notificationURI:http:
//localhost:9999 specName: specCURRENT
```

The ALE will start sending empty ECReports to the event sink GUI as the Rifidi emulator has no tags in field yet.

Set tags in the reader field Now we will make the virtual reader start delivering tags to the ALE. From the

"TagsView" of Rifidi emulator, create a new tag (SGTIN96, GEN2) then drop it into the antenna zone. The tag reads will be reported to the Fosstrak ALE Middleware. After the tag reads are filtered and collected as specified in the ECSpec, they are delivered by the Fosstrak ALE middleware to the event sink GUI.

Based onFosstrakALE MiddlewareUser guide.

Developers Guide

Acquiring the code

You can get the source code of ALE through Aspire SVN. You should get all the source code under the path aspire/trunk/servers/AspireRfidALE/ALE. Building

You will need to:

- install Maven 2.2
- install subversion
- install java runtime

Then :

- svn checkout svn://svn.forge.objectweb.org/svnroot/aspire/trunk/servers/AspireRfidALE/ALE
- download the following dependencies :

javax.comm

eclipse webserviceUtils (needed only by fc-webclient)

ltkjava-1.0.0.7 updated library (needed by fc-server)

- Install the dependencies locally :

```
mvn install : install-file -Dfile=< path-to-file > -DgroupId=< group-id > -DartifactId=<
artifact-id >
```

For webserviceUtils : groupId = org.eclipse.jst.ws.consumption, artifactId = webserviceutils, version = 1.0.102.v200609220223

- mvn package

Aspire :: Filtering & Collection (en)
Creator: xwiki:XWiki.nkef Date: 2009/01/29 17:57
Last Author: xwiki:XWiki.driss_r Date: 2014/12/04 14:23