ALIEN TECHNOLOGY®

ON-READER PROGRAMMING OVERVIEW

Ruby API for

Alien RFID Custom Application (ARCA™) Kit

January 2016

All Readers

Legal Notices

Alien Technology®

Alien ARCA™ On-Reader Programming Overview

(this page was intentionally left mostly blank)

Introduction

This document describes how to create, load, and run applications on Alien Technology RFID readers, using the Alien RFID Custom Application ARCA™, which features the Ruby programming language. Supported readers include:

- ALR-F800
- ALR-9900+
- ALR-9900
- ALR-9650
- ALR-9800
- ALR-8800

Background

Alien readers communicate via the Alien Reader Protocol, a flexible interface featuring AutoMode, notification, tag masking, tag and IO-streaming and other powerful features. Coupled with the Alien Reader Protocol, ARCA provides a feature-rich, highly adaptable platform for implementing RFID solutions. Now users can easily implement custom extensions to the Alien Reader Protocol, including RFID data pre-processing, application-specific triggering modes or custom communication channels with server-side applications.

ARCA expands the utility of Alien readers in ways that only our inventive users can predict. On-reader applications can eliminate the need for a centralized reader management system so that readers can react to events autonomously. Many remote and mobile applications will benefit from this capability. With ARCA users can offload business logic to readers to distribute decision making and enhance system reliability. On-reader applications enable a wide range of custom filtering, pre-processing and data formatting options. The Alien RFID Custom Application Kit enables users to discover more value, more quickly, for applications like retail inventory management, industrial manufacturing and asset tracking.

Why Ruby? Alien selected Ruby because it is a stable, flexible, object-oriented programming language that allows users to rapidly develop cross-platform applications. Ruby is relatively easy to master. It is suitable to the scale of applications RFID users want to implement on our readers. Coupled with the Alien Reader Protocol, Ruby provides an easy to use, feature-rich, highly adaptable platform for implementing RFID solutions.

Quick Start

- Install Ruby and the Alien Ruby API on the reader.
- Learn to use Ruby and the Alien Ruby API.
- Look at API Examples, write an Application
- Use RAI utility to manage applications on the reader.

Install Ruby & Alien Ruby API on the Reader

The Ruby Interpreter

Running Ruby applications on the reader requires the Ruby interpreter (includes Ruby standard libraries), the Alien Ruby RFID API, and your application source files. Ruby source files are plaintext and interpreted by Ruby on-the-fly. No compilation or processing of your source code is required.

ALR-F800

The ALR-F800 readers already come with the Ruby interpreter installed!

ALR-9900+, ALR-9900, ALR-9680, ALR-9650, ALR-9800, ALR-8800

For these older readers, you mush first install the Ruby interpreter on your reader using the Alien firmware package that has a filename similar to:

alien-ruby_x.x.x-1_arm.tar.aef

Use the reader's web interface to **install the Ruby interpreter onto the reader**, as you might install a firmware upgrade. Go to your reader's IP address using a web browser, click on Reader Management, log in, then point to the `alien-ruby_x.x.x-1_arm.tar.aef` file and click on "Upload" to install the package.

The Alien Ruby RFID Library

In addition to the Ruby interpreter, the Alien Ruby RFID libraries must also be on your reader. Your application uses these library files to communicate with the reader, implementing the full set of Alien Reader Protocol commands as well as parsing tag data, listening on sockets, etc.

All Alien RFID readers usually come with the Alien Ruby RFID libraries already pre-installed. Follow the steps below to verify that and, if necessary, **install the Alien Ruby RFID libraries onto the reader**. You do this the same way you would install new firmware using the reader's web interface.

ALR-F800

Verify that the Alien Ruby RFID libraries are installed on your ALR-F800 reader. Use a web browser to go to your reader's IP address, click on the *Administration* > *Logs* navigation link and log in if prompted. Select "Alien Packages" from the log selector and look to see if "alien-ruby-api" is in the list of installed packages.

If you find that you do need to install or upgrade your libraries, click on the *Administration* > *System* navigation link, then click on the *Browse* button in the Upgrade area of the screen and direct it to the `alien-ruby-api` installer package, then click the *Install* button. The installer should have a filename similar to

alien-ruby-api_x.x.x_armel.aed

ALR-9900+, ALR-9900, ALR-9680, ALR-9650, ALR-9800, ALR-8800

Verify that the Alien Ruby RFID libraries are installed on your reader. Use a web browser to go to `http://<readerIP>/cgi-bin/logs.cgi` and log in if prompted. Select "DEB Alien Packages" from the log selector and look to see if "alien-ruby-api" is in the list of installed packages.

If you find that you do need to install or upgrade your libraries, use your web browser to go to `http://<readerIP>/cgi-bin/manage.html`, and use the "Upload a Firmware Update" area to upload the alien-ruby-api installer package. The installer should have a filename similar to

alien-ruby-api_x.x.x-1_arm.tar.aef

Learn to Use Ruby and the Alien Ruby API

Familiarize Yourself with Ruby

While Ruby is very intuitive and flexible, some aspects of it are significantly different from other programming languages. We recommend that those who are new to Ruby spend some time at **http://www.ruby-lang.org**. This site offers an interactive Ruby tutorial, documentation, forums, and tips on transitioning to Ruby from other languages. Information about Ruby installers for other platforms is also available at this web site.

The version of Ruby installed on readers is 1.9.3 on ALR-F800 readers, and 1.8.7 on older readers. In order to avoid compatibility issues when developing Ruby applications for Alien readers on your PC make sure to install on your PC a Ruby version that is similar to the reader's version.

The best place to get Ruby distributions and installation instructions is at

https://www.ruby-lang.org/en/documentation/installation/

On Windows platforms, the easiest option to install Ruby is to use a Ruby one-click installer from:

http://rubyinstaller.org/downloads/archives

On Linux, there are several ways to install Ruby:

- some Linux distributions already include Ruby.
- you can also download the source code and compile it by hand.
- on some platforms, there are package management solutions that make installing Ruby extremely easy. For example, you can use 'apt-get' on Debian or Ubuntu:
  ```
  sudo apt-get install ruby irb rdoc
  ```

The distribution of Ruby that is on the reader includes only the standard Ruby libraries. No additional extensions, known as Ruby "gems" have been installed. There are no plans to support gems on older readers but they will be supported on the ALR-F800 and newer readers.

Before you start creating reader applications, use the `ruby-lang.org` website to become proficient enough at Ruby to work through the tutorials, and even write your own apps from scratch. Get comfortable running Ruby applications from your system's command-line. If you used the "one-click-installer" to install Ruby on your Windows system, it will automatically associate the `.rb` extension with Ruby, and add the Ruby executable to your command path.

Next, begin examining the lib directory inside the `alien-rfid-lib` folder and study the class files Alien has put together to communicate with your RFID reader.

alien-rfid-lib

The library files in the `alien-rfid-lib/` folder are exactly the same as those that you previously installed on the reader. In fact, when running your Ruby application on the reader, the only difference in your code is the IP address used to connect to the reader. When running your application on the reader, you simply use `127.0.0.1` (or `localhost`) for the reader's IP address. We also recommend that you connect to the `CommandPortLocal` (`2300`, by default). This port is intended to support on-reader applications. It is an independent command channel, just like the normal, port 23 command channel, but it only accepts connections originating from within the reader itself. Using port 2300 leaves the standard port 23 available for external connections to the reader, so reader can be controlled from both the on-reader application and the command channel on port 23 at the same time. This could be desirable or not depending on the logic of your application, so select whichever port works for your use case.

You can **put the `alien-rfid-lib/` folder wherever you want on your PC's hard drive**. The only requirement is that you should stage your RFID applications in your own folder inside `alien-rfid-lib/`, next to the `lib/` folder, as shown below. The same folder structure is maintained on the reader to assure that your application will find the required Alien libraries when running on the reader.

```
alien-rfid-lib/
  |---  lib/
  |---  examples/
  |---  yourApp1/
        |---  yourApp1.rb
        |---  other files1
  |---  yourApp2/
        |---  yourApp2.rb
        |---  other files2
```

Inside the `lib/` folder there is a set of Ruby classes written by Alien whose use is demonstrated in the files inside the `examples/` folder. You have full access to the source code for these classes, and you may find that looking through them improves your understanding or Ruby and how the classes work, but you should refrain from modifying the library files directly.

Please forward bug reports and change requests for these libraries directly to Alien for incorporation into future versions of the Alien Rubi API. The library files residing on the reader are maintained by Alien, via the *alien-ruby-api_x.y.z-1_arm.tar.aef* package installer.

The reader identifies your application by the same name as the directory that the application resides in, so the ***main file for your application must have the same name as the directory name***. Thus, an application named "foobar" would be located at `alien-rfid-lib/foobar/foobar.rb`.

All file and folder names should conform to Linux file naming standards. Also, avoid using spaces and subdirectories. You can use multiple Ruby source code files in your project but they all have to reside **in the same folder** – in your application folder. This is because the current version of RAI tool only supports flat folder structure and does not support subfolders.

When your application needs to reference the Alien classes in the `lib/` folder, your use a relative reference, like "`../lib/alienreader.rb`". When your application is finally installed and run on the reader, it is copied into a similar directory alongside the same `lib/` folder.

Alien RAI (Reader Application Interface) for your PC

In addition to the *alien-rfid-lib* API and associated reader packages, you also need to install the Alien Reader Application Interface (RAI) tool on your PC. This utility facilitates loading and execution of user applications on the reader. The RAI tool is a Ruby script called `rai.rb`, and is included in the SDK distribution.

You need to be able to invoke the RAI tool without too much fuss, and from any arbitrary application directory, so we recommend that you **place `rai.rb` in a directory that is in the PATH on your PC**, so that the system can find the utility no matter what your current working directory is. If you do this, and your PC also associates the `.rb` extension with the Ruby interpreter, then you can run the RAI tool from the command prompt from any directory just by entering the command "`rai`".

If the `.rb` extension is not associated with the Ruby interpreter, you will have to type something less elegant, like "`ruby <path_to_rai.rb>\rai.rb`"

More information on the usage of RAI tool is provided in a later section, "Use RAI to Manage Apps on the Reader".

net-scp

The RAI tool uses Secure Copy (SCP) to transfer files to your reader and issue commands remotely. The default Ruby installation generally doesn't include the required SCP and SSH libraries. The easiest way to **install the `net-scp` Ruby "gem"** is to use Ruby 'gem' distribution mechanism.

Note: As mentioned above, Ruby gems are not supported on readers. For this reason we do not generally recommend installing them and relying on their functionality on your Ruby-RFID development PC. However, this gem is an exception that simplifies the installation of '`net-ssh`' and '`net-scp`' libraries used by the Alien RAI tool and we do find using gem mechanism useful in this case.

Make sure to install ‘`net-ssh’` gem **first** and then install ‘`net-scp’` gem.

The version of ‘`net-ssh’` gem to be installed depends on the version of Ruby installed on your PC. To check the installed version of Ruby issue the following command **at the command prompt**:

```
ruby -v
```

The following instructions presume that your PC is connected to the Internet. If not, then see the following paragraph on how to install gems manually.

If you have **Ruby 1.8.x** or **Ruby 1.9.x** then install ‘`net-ssh’` gem version 2.9.2

```
gem install net-ssh –v “=2.9.2”
gem install net-scp
```

If you have **Ruby 2.0** or later then install any current release of ‘`net-ssh’` gem (v3.0.2 or later).

```
gem install net-ssh
gem install net-scp
```

Alternatively, and in cases when internet connection is not available, you can install the required gem packages locally, without resorting to the automated gem installation method:

- The required gem packages are in the Alien Ruby SDK or you can download them:

'`net-ssh`' package:	
Ruby 1.8.x or 1.9.x:	https://rubygems.org/downloads/net-ssh-2.9.2.gem
Ruby 2.0 or later:	https://rubygems.org/downloads/net-ssh-3.0.2.gem
'`net-scp`' package:	https://rubygems.org/downloads/net-scp-1.2.1.gem

- Copy the .gem package files to any folder on your PC and install them as follows:
  cd \path\to\gem

'net-ssh' package:	
Ruby 1.8.x or 1.9.x:	gem install --local net-ssh-2.9.2.gem
Ruby 2.0 or later:	gem install --local net-ssh-3.0.2.gem
'net-scp' package:	gem install --local net-scp-1.2.1.gem

The only requirement for NET::SSH2 you might be missing is the OpenSSL bindings for Ruby. These are built by default on most platforms (the one-click installer for Windows includes them) and to verify that they're installed on your system just run the following on the command line:

```
ruby -ropenssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

You should see something like "OpenSSL 1.0.11 15 Jan 2015". If you get an error, then you'll need to either rebuild Ruby with OpenSSL support, or, if your platform supports it, install the OpenSSL bindings separately. For example, on Debian or Ubuntu you can use 'apt-get':

```
sudo apt-get install libopenssl-ruby
```

API Documentation

The Alien Rubi API documentation is created with a tool called *rdoc*, which works similarly to *JavaDoc*, by scanning specially-formatted comments in the source code and generating a suite of HTML files. You can view API and examples documentation with your web browser at:

Documentation/alien-rfid-lib/index.html

Many examples of usage are available in the documentation, in addition to the set of Ruby example files that Alien provides.

If you examine the documentation on the *AlienReader* class, you may be surprised to see only a handful of methods defined to issue reader commands. The bulk of the code that handles the Alien Reader Protocol (ARP) command set is generated on-the-fly when the *AlienReader* class is created. It scans an external `readermethods.dat` file which lists all of the reader commands and their get/set/do usage. From each command description in `readermethods.dat`, the class builds accessor methods with the same names as the command – the ARP command name in lowercase letters.

For instance, to get the *AntennaSequence*, use the getter method:

```
AlienReader.antennasequence
```

To set the *AntennaSequence*, use the setter method:

```
AlienReader.antennasequence="0 1"
```

Some commands are "do" kinds of commands, such as the method:

```
AlienReader.automodereset
```

Look at API Examples, Write an Application

API Examples

Inside the `examples/` folder there are over a dozen sample Ruby applications, each demonstrating different aspects of the API classes. For a quick overview of the files and what they do, just view the documentation for the examples, at

Documentation/examples/index.html

config.ini

In addition to the sample `.rb` files, there is one other important file in the examples directory, `config.ini`. We recommend that you create your own `config.ini` (or copy this one) for use in your application. It resides in your application directory. It is a preference file that contains IP addresses, ports, username/password, and any other configuration information that your application might need.

You don't have to use `config.ini`. You can handle your own application preferences and settings in other ways, but this is how the examples work, so it's worth becoming familiar with it. By storing common configuration information in one file, it is easily accessible to all of your Ruby scripts by using the provided *AlienConfig* class. You'll see how it is used in the examples to follow.

The first entry in `config.ini` is generally the IP address of the reader your application needs to connect to. When you transfer your Ruby application to the reader you just need to change the IP address to `127.0.0.1` (`localhost`) and your app should work the same as when it was running remotely.

ex_connect.rb

This application will connect to a reader (with IP address specified in `config.ini`) get its *ReaderName*, and print a "Hello world!" greeting from the reader.

```
=begin rdoc
=Alien Ruby RFID Library Examples
==ex_connect.rb
Once you have a reader's IP address, you can to connect to it. This example shows how
to make a connection to a reader, login, and query the reader for basic information.

Uses the AlienReader.open command with default values for port, username and password.

Change the "reader_address" parameter in your config.dat file to the one appropriate
for your reader.

Copyright 2014, Alien Technology Corporation. All rights reserved.
=end

# add the default relative library location to the search path
$:.unshift File.join(File.dirname(__FILE__), '..', 'lib')

require 'alienreader'
require 'alienconfig'

```

```
19  begin
20  # grab various parameters out of a configuration file
21    config = AlienConfig.new('config.ini')
22
23  # change "reader_address" in the config.ini file to the IP address of your reader.
24    ipaddress = config.fetch('reader_address', 'localhost')
25
26    r = AlienReader.new
27
28    puts '-----------------------------------'
29
30    if r.open(ipaddress)
31      puts 'Hello World!'
32      puts "My name is: #{r.readername}."
33      puts "I am an:    #{r.readertype}"
34    end
35
36    puts '-----------------------------------'
37
38    # be nice. Close the connection to the reader.
39    r.close
40  rescue
41    puts $!
42  end
```

Lines 1-11	=begin…=end is a multi-line commend. The contents are formatted so that rdoc can use them.
Line 14	$: is the variable containing the library path. We are prepending "../lib" to the path to make the Alien classes available. This is more complicated than it probably needs to be.
Lines 16-17	These two lines link in the AlienReader and AlienConfig classes. AlienConfig allows us to use the data in the config.ini file, and AlienReader provides the class to communicate with the reader.
Lines 19,40, 42	A begin-rescue-end block is the equivalent of try-catch block. The code before "rescue" is evaluated until an exception is raised, then the code under "rescue" is run.
Line 21	Creates a new AlienConfig object, which reads and parsed the contents of config.ini. The resulting config variable is a hash, allowing us to access any of the fields in config.ini by name.
Line 26	Creates a new AlienReader object.
Lines 28, 31	Puts prints text to the screen.
Line 30	Opens a connection to the reader, using the IP address given by the config hash. This version of AlienReader.open() uses the default username, password, and commandport values. If the connection is made, open() returns true.
Lines 31-33	Prints the "Hello World" string to the screen. Embedded one of the strings is #{r.readername}, which is a method call, "readername" on the AlienReader object, r. The surrounding #{} structure is a way of embedding Ruby code directly in a string literal. We could have also written: puts "My name is: " + r.readername + "."
Line 39	Closes the connection to the reader.

RUNNING EX_CONNECT.RB

Make sure the config.ini file in the examples/ folder has the correct connection information for your reader, then type "**`ruby ex_connect.rb`**" at your PC's command-line. You should see:

```
$ cd /<pathToYourRubyAPI>/alien-rfid-lib/examples
$ ruby ex_connect.rb
----------------------------------
Hello World!
My name is: David's F800.
I am an: ReaderType = Alien RFID Tag Reader, Model: ALR-F800 (902-928 MHz)
----------------------------------
$
```

Let's look at a more complicated example.

ex_get_taglist.rb

This example connects to the reader, gets a taglist, and parses it into an array of *AlienTag* objects. It then sorts them based on their EPC codes and prints the list.

```
=begin rdoc
=Alien Ruby RFID Library Examples
==ex_get_taglist.rb
An example program to play with taglists.

* Connect to an Alien RFID reader. Login.
* Grab some tag data.
* Scan the data for interesting tags and display the results.

Copyright 2014, Alien Technology Corporation. All rights reserved.
=end

# Add the default relative library location to the search path
$:.unshift File.join(File.dirname(__FILE__), '..', 'lib')

require 'alienreader'
require 'alientag'
require 'alienconfig'

begin
  # Grab various parameters out of a configuration file
  config = AlienConfig.new('config.ini')

  # Change "reader_ip_address" in config.ini to the IP address of your reader.
  ipaddress = config.fetch('reader_address', 'localhost')

  # Create a reader
  r = AlienReader.new

  # Create a tag list
  tl = Array.new

  if r.open(ipaddress)
    puts '----------------------------------'
    puts "Connected to: #{r.readername}"

    # Construct a taglist from the reader's tag list string
```

```
38      # Note: if automode is running this will contain the latest tags.
39      # If not, the reader will read tags and then return the data.
40      tl = AlienTagList.new(r.taglist)
41
42      # How many tags did we find?
43      puts
44      puts "Number of tags found: #{tl.length}"
45
46      # Sort your list to make reading easier. The comparison operator <=>,
47      # used by sort, is part of the Tag class in alientag.rb.
48      tl.sort!
49      puts tl
50
51      # Did we find a particular tag? You can use a regular expression to check
52      # if elements in the list are tags that match what you are interested in.
53
54      puts 'Tag List Matches:'
55      puts "Tag #\tTag ID"
56      puts tl.filter(/.*/) # use a regex to filter specific tags
57      puts "---------------------------------"
58
59      # Be nice. Close the connection.
60      r.close
61    end
62  rescue
63    puts $!
64  end
```

Lines 1-11	=begin…=end is a multi-line commend. The contents are formatted so that rdoc can use them.
Line 14	$: is the variable containing the library path. We are prepending "../lib" to the path to make the Alien classes available. This is more complicated than it probably needs to be.
Lines 16-18	These lines link in the AlienReader, AlienTag, and AlienConfig classes.
Line 20-35	This is the main part of the program – the part that is executed first. We use AlienConfig to pull the reader's IP address out of config.ini, prepare an array to hold our tag data, create the reader object, and open a connection to it.
Line 40	Fetches the taglist from the reader (r.taglist), and creates a new AlienTagList out of it.
Line 44	Prints the number of tags, and dumps the array (p is a shortcut to puts).
Line 48	Sorts the taglist array. The ! after the sort method name indicates the sort method modifies the object itself, instead of just returning a sorted copy.
Line 49	Dumps the taglist array.
Line 56	Filters the tags using a very loose /.*/ regular expression (it matches anything). The filtered output is printed to the screen.
Line 60	Closes the connection to the reader.

RUNNING EX_GET_TAGLIST.RB

Make sure the config.ini file in the examples/ folder has the correct connection information for your reader, then type "**`ruby ex_get_taglist.rb`**" at the command-line. You should see:

```
$ cd /<pathToYourRubyAPI>/alien-rfid-lib/examples
$ ruby ex_get_taglist.rb
----------------------------------
Connected to: David's 9800
Number of tags found: 16
[0000 0000 0000 0000 0000 0008, 0000 0000 0000 0000 0000 0005, 0000 0000 0000 0000 0000
    0007, 0000 0000 0000 0000 0000 0006, 0000 0000 0000 0000 0000 000A, 0000 0000 0000
    0000 0000 0001, 0000 0000 0000 0000 0000 0004, B00B 1111 0000 0000 0000 0000, 0000
    0000 0000 0000 0000 000D, 0000 0000 0000 0000 0000 000C, B00B 0000 0000 0000 0000
    0000, 5555 1111 1111 1111 1111 1102, 0000 0000 0000 0000 0000 000B, 0000 0000 0000
    0000 0000 0003, 0000 0000 0000 0000 0000 0002, 0000 0000 0000 0000 0000 0009]
Tag List Matches:
Tag # Tag ID
0     0000 0000 0000 0000 0000 0001
1     0000 0000 0000 0000 0000 0002
2     0000 0000 0000 0000 0000 0003
3     0000 0000 0000 0000 0000 0004
4     0000 0000 0000 0000 0000 0005
5     0000 0000 0000 0000 0000 0006
6     0000 0000 0000 0000 0000 0007
7     0000 0000 0000 0000 0000 0008
8     0000 0000 0000 0000 0000 0009
9     0000 0000 0000 0000 0000 000A
10    0000 0000 0000 0000 0000 000B
11    0000 0000 0000 0000 0000 000C
12    0000 0000 0000 0000 0000 000D
13    5555 1111 1111 1111 1111 1102
14    B00B 0000 0000 0000 0000 0000
15    B00B 1111 0000 0000 0000 0000
----------------------------------
```

Your Application

Now look at the other examples, play with them, and try some stuff out. More importantly, think of a problem or use-case that you'd like to solve and try to implement a solution!

Remember, right now you are still running Ruby code on your PC, and connecting to the reader over an external socket. When you install your application on the reader, the exact same code will work the same, except that you'll be connecting over a local socket on the *CommandPortLocal* (2300).

Use RAI Utility to Manage Apps on the Reader

In order to conveniently and safely install your application on the reader, you should use the RAI (Reader Application Interface) utility. The RAI utility is actually a Ruby script called `rai.rb` **that needs to be copied to a directory that is in the PATH on your PC** to ensure that it is easy and convenient to use it.

The RAI utility does the job of installing application files onto the reader, retrieving files, listing the applications installed on the reader and the files contained in each one, registering your application as a reader "service" and getting the current status of installed applications.

Here is a walkthrough of RAI in action, using one of the supplied example files, `ex_io_ant.rb`. This sample program looks at the reader's *ExternalInput* value and, if it is non-zero, sets the *AntennaSequence* using the *ExternalInput* as a bitmask (each bit enables an antenna) and then finally turns *AutoMode* on. This is a good example because you can look at the antenna LEDs on the reader to see what it is doing.

If you don't have the hardware to actuate the *ExternalInputs*, we'll show you how to modify the program to read the *ExternalOutputs* instead (which you can change manually at the Alien> prompt).

Create and Run ioant on the PC

1. Make a new application directory, called "`ioant`". That'll be the name of our program.

```
C:\>cd \ruby\alien_rfid_lib

C:\ruby\alien_rfid_lib>dir
Directory of C:\ruby\alien_rfid_lib

10/31/2008  04:10 PM    <DIR>   .
10/31/2008  04:10 PM    <DIR>   ..
12/17/2008  03:16 PM    <DIR>   examples
11/18/2008  10:35 AM    <DIR>   lib
              1 File(s) 289 bytes
              4 Dir(s)  12,084,957,184 bytes free

C:\ruby\alien_rfid_lib>mkdir ioant

C:\ruby\alien_rfid_lib>cd ioant
```

2. Copy the example program into the new ioant folder, changing the name from `ex_io_ant.rb` to just `ioant.rb`. Also, copy `config.ini` out of the examples folder.

```
C:\ruby\alien_rfid_lib\ioant>copy ..\examples\ex_io_ant.rb ioant.rb
      1 file(s) copied.

C:\ruby\alien_rfid_lib\ioant>copy ..\examples\config.ini .
      1 file(s) copied.
```

3. Modify `config.ini` for our setup. I removed everything except for the connection information.

```
C:\ruby\alien_rfid_lib\ioant>e config.ini
(editing file)

C:\ruby\alien_rfid_lib\ioant>type config.ini
#Network Parameters for Reader
reader_ip_address = 10.10.82.72
port = 23
username = alien
password = password
```

4. (Optional) If you don't have the ability to flip digital inputs connected to your reader, you can modify the `ioant.rb` file to look at *ExternalOutputs* instead, which you can change manually via a serial connection to the reader or the web interface. Also, instead of turning *AutoMode* on (which might reset the *ExternalOutputs* to 0 while it is running), you can just do repeated "*get taglist*" calls instead.

Modify `ioant.rb` so that the main code block looks like this (changes are in bold):

```
. . .
if r.open(ipaddress)
    puts '-----------------------------------'
    puts 'Connected to ' + r.readername

    #read input and init variables
    dig_in = r.externaloutput.to_i                  # look at outputs instead
    old_dig_in = dig_in

    done = false

    # spin here forever... (or ctrl-c)
    until done
      dig_in = r.externaloutput.to_i                # look at outputs instead

      if dig_in != 0                                # Don't do anything if XO=0
        if dig_in != old_dig_in                     # Change AntSeq if XO changed
          r.antennasequence = map_input_to_antennas(dig_in)
        end
        r.taglist                                   # Fetch taglist (no AutoMode)
      end

      old_dig_in = dig_in
      sleep 0.1                                     # sleep 0.1 sec, not 1 sec
    end

    puts '-----------------------------------'

    # be nice. Close the connection to the reader.
    r.close
end
. . .
```

5. Run `ioant.rb` locally. While it is running, either change your digital inputs, (or outputs if you modified the example) and you should see the corresponding antenna LEDs lighting up. Type Ctrl-C to exit (this may leave AutoMode running).

```
C:\ruby\alien_rfid_lib\ioant>ruby ioant.rb
----------------------------------
Connected to David's 9800

(ctrl-c)
ioant.rb:77:in `sleep': Interrupt
      from ioant.rb:77
```

Install the ioant Application on the Reader

Now we'll use the RAI tool to load the *ioant* application onto your reader. Make sure RAI has been properly installed by running "`rai help`" at the command prompt.

```
C:\ruby\alien_rfid_lib\ioant>rai help

Alien Reader Application Interface (RAI) ver.1.0.0

Usage: rai.rb [options]

Configure options:
    configure (conf,config)      configure authorization options

Transfer options:
    put [<file1>[ <file2>...]]   put file(s) to the reader
    get [<file1>[ <file2>...]]   retrieve file(s) from the reader

Install options:
    register (rg, reg)           register application on the reader
      -p, --priority <priority>  application startup priority (70-90), default 80
      -c, --cli <cli>            application Command Language Interpreter (ruby, bin)
    unregister (ur, ureg, unreg) unregister application on the reader
    delete (del)                 delete application on the reader

Status options:
    list (l)   [apps|<app>]      list applications installed on the reader
    status (s) [apps|<app>]      display status of application on the reader

Common options:
    help (h)                     display help information
    version (v, ver)             display version information
```

Before RAI can transfer files to your reader, you need to configure it with the reader's IP address, login credentials, the command interpreter for your app (ruby), and the default startup priority (in relation to other reader services).

1. Run "**`rai conf`**" to configure RAI. Use your reader's IP address, a username of "`alien`", and the following password: “password” (ALR-F800) or "`a113n`" (for all other readers). Your command line interpreter (cli) is "ruby”. A startup priority of 80 will work in most cases.

```
C:\ruby\alien_rfid_lib\ioant>rai conf
Enter 'hostname': 10.10.82.72
Enter 'username': alien
Enter 'password': a113n
Enter '     cli': ruby
Enter 'priority': 80
```

This generates a "`.rai`" configuration file in your application directory. Once this configuration file is created RAI won't have to keep asking you for this information. You can rerun "`rai conf`" at any time to edit this file.

2. Before you load the files, you need to change `config.ini` so that when *ioant* runs on the reader, it connects to *localhost* (127.0.0.1) on port 2300.

```
C:\ruby\alien_rfid_lib\ioant>e config.ini
(editing file)

C:\ruby\alien_rfid_lib\ioant>type config.ini
#Network Parameters for Reader
reader_ip_address = 127.0.0.1
port = 2300
username = alien
password = password
```

3. Now load the entire set of application files with "**`rai put`**" (you must do this from within the `ioant/` directory).

```
C:\ruby\alien_rfid_lib\ioant>rai put
uploading 'C:/ruby/alien_rfid_lib/ioant/config.ini'...done
uploading 'C:/ruby/alien_rfid_lib/ioant/ioant.rb'...done
```

At this point, your files are loaded into the proper location on the reader, right alongside the Alien API libraries. In order to run *ioant* (manually or automatically after the reader boots up), you need to register it with the reader subsystem.

4. Verify the files were successfully transferred to the reader with "**`rai list`**".

```
C:\ruby\alien_rfid_lib\ioant>rai list
config.ini
ioant.rb
ioant.sh
```

Register the ioant Application and Run It

Once your application is loaded on the reader, you register it with the "**`rai register`**" command. This generates a control script for your application, and links that control script into the reader subsystem along with the other reader services, such as the Heartbeat, SNMP, Serial interface, and the network Interface Monitor (*ifmon*).

You can then use the Alien Service command at the *Alien>* prompt to manage your app.

1. Register the application on the reader with "**`rai register`**". Since this generates files on the reader and changes the way the reader might operate, you are asked to confirm this action.

```
C:\ruby\alien_rfid_lib\ioant>rai register
Register 'ioant' RUBY application as Alien service with PRIORITY=80?
    please type 'yes' or 'no':yes
```

2. Verify that ioant is installed and registered by asking RAI for the list of applications: "**`rai list apps`**". The "r" prefix indicates the application has been registered as a reader service.

```
C:\ruby\alien_rfid_lib\ioant>rai list apps
r ioant
```

3. Check the status of the application/service with "**`rai status`**".

```
C:\ruby\alien_rfid_lib\ioant>rai status
s A 80 ioant
```

4. Notice that this looks just like what you see when you use the *Service* command at the *Alien>* prompt (I'm doing this over the serial interface).

```
Alien>Service all status
R A 10 serial
R A 20 snmp
R A 40 heartbeat
s A 80 ioant
R A 95 ifmon
```

5. The "s" indicates the service is ***s***topped ("R" indicates ***R***unning). The next letter, "A", indicates the service is configured to ***A***utostart after the reader boots (a "d" indicates this is ***d***isabled). Therefore, our new *ioant* service is not running, but will start up whenever the reader boots. Let's start it manually at the *Alien>* prompt with "`Service ioant start`", verify it works, and then stop it and disable autostart.

```
Alien>Service ioant start
R A 80 ioant

Alien>ExternalOutput=55 (or actuate your digital inputs)
```

```
ExternalOutput = 5
(verify the reader is reading on antennas 0 and 2)

Alien>Service ioant stop
s A 80 ioant

Alien>Service ioant disable
s d 80 ioant
```

Unregister the ioant Application

If you no longer want your application to be available as a reader service, you can unregister it with "**`rai unregister`**".

```
C:\ruby\alien_rfid_lib\ioant>rai unregister
Unregister 'ioant' application from being an Alien service?
    please type 'yes' or 'no':yes

C:\ruby\alien_rfid_lib\ioant>rai list apps
    ioant
(no longer has "r")

Alien>Service all status
R A 10 serial
R A 20 snmp
R A 40 heartbeat
R A 95 ifmon
(missing from list of Services)
```

Retrieve Application Files from the Reader

If you want to retrieve application files from the reader – log files, data files, etc. you can use "**`rai get`**". Without any further arguments, the get command retrieves all of the files in the application directory, or you can indicate which file(s) to get. This action will **overwrite** any similarly named files in the current application directory!

```
C:\ruby\alien_rfid_lib\ioant>rai get
downloading 'config.ini'...done
downloading 'ioant.rb'...done
downloading 'ioant.sh'...done

C:\ruby\alien_rfid_lib\ioant>rai get config.ini ioant.rb
downloading 'config.ini'...done
downloading 'ioant.rb'...done
```

Remove an Application from the Reader

To completely remove your application files from the reader, use the "**`rai delete`**" command.

```
C:\ruby\alien_rfid_lib\ioant>rai delete
Delete 'ioant' application from the reader?
    please type 'yes' or 'no':yes

C:\ruby\alien_rfid_lib\ioant>rai list apps
```

```
(nothing)
```

Working with Removable Media (ALR-F800)

The Alien ALR-F800 reader supports removable media, including USB flash drives and micro-SD cards. This media can be used by on-reader Ruby applications to read or store data or logs to/from the removable media.

For added convenience, USB flash drives and micro-SD cards are automatically mounted to the `/media` folder on the reader as soon as they are inserted.

If the removable media has a label (volume name) then it will be accessible at `/media/<LABEL>`. For example, an SD card labeled as “MYCARD” will be mounted at `/media/MYCARD`.

If the media has no label set, then it is mounted to its device name location, usually `/media/mmcblk1p1` for micro-SD cards and `/dev/sda1` for USB flash drives.

Supported file system formats are FAT, FAT32, Ext3, Ext4. It is recommended that you use the native Ext4 Linux file system format as it provides the best data protection and performance on the reader.

WARNING: As with any flash media, frequent random writes and flush operations can seriously increase a flash device’s I/O latency and reduce the device lifetime.

The following code fragment shows how to fetch reader configuration from a file loacated on an USB flash disk labeled as ‘USB_DISK’.

```
begin
    config = AlienConfig.new('config.ini')
    addr = config['rdr_addr']
    port = config['rdr_port']

    r = AlienReader.new

    if r.open(addr, port)

      rdrconf = '/media/USB_DISK/rdr.conf'

      if File.file?(rdrconf)
        File.open(rdrconf).each do |line|
          r.sendreceive(line)
        end
      end

#     ...

      r.close
    end
rescue
    puts $!
end
```

Accessing Serial Ports from Ruby Apps

Support for using the serial interface is included in the Alien ARCA Ruby distribution. If you want to use the serial port interface from your Ruby app, you need to disable both Alien and Linux command prompt serial interfaces.

Disable Alien Serial Interface

The Alien serial port interface can be stopped and disabled as follows.

```
Alien>service serial disable
Alien>service serial stop
```

NOTE: if you issue these commands while using Alien serial interface, make sure to ‘disable’ first and then ‘stop’. If you issue the ‘stop’ command first, then you won’t be able to issue the ‘disable’ command. You can issue those commands over a TCP connection in any sequence.

Disable OS Serial Interface

ALR-F800

Either serial interface (RS-232 or USB) can be completely release with a service command. To release both serial ports:

```
Alien>service serial release
```

To only release the RS-232 (serial0) or USB (serial1), use one of those specific serial addresses:

```
Alien>service serial0 release
Alien>service serial1 release
```

ALR-9900+, ALR-9900, ALR-9680, ALR-9650, ALR-9800, ALR-8800

In order to disable the Linux serial interface on these older readers, you have to create the following file on the reader:

```
/home/alien/alien.notty
```

Use putty or a similar terminal program to login into Linux as:

```
username: alien
password: a113n
```

The easiest way to create a file is to issue the following command from the Linux prompt:

```
touch /home/alien/alien.notty
```

Reboot the reader and after the Linux boots up, you should have a serial port available for your Ruby application.

Serial Libraries in Ruby

ALR-F800

ALR-F800 readers come pre-installed with the “`serialport`” Ruby gem. Simply add the ‘`require serialport`’ directive in your Ruby apps to link in those libraries.

ALR-9900+, ALR-9900, ALR-9680, ALR-9650, ALR-9800, ALR-8800

On these readers, the “serialport” libraries are built into the Alien ARCA Ruby distribution. Do NOT include the 'require serialport' directive in your Ruby apps running on the reader because the serial library is already compiled into the reader's 'ruby' interpreter.

Serial Port Example

```ruby
# Add this line only on the ALR-F800:
require 'serialport'

begin
    # ALR-F800 RS-232: use /dev/ttyPS0
    # ALR-F800    USB: use /dev/ttyPS1
    # Other Readers  : use /dev/ttyS0
    sp = SerialPort.new('/dev/ttyPS1', 115200, 8, 1, SerialPort::NONE)

    # Write to the serial port:
    sp.write "Hello\r\n"

    # Read until <CR>, repeat until receive 'exit':
    loop do
      s = sp.readline("\r")
      puts "received #{s.upcase}"
      break if s.downcase.strip == 'exit'
    end

    # Close port:
    sp.close
rescue
    puts $!
end
```

For more information about the serialport interface, refer to these resources:

http://ruby-serialport.rubyforge.org/

http://ruby-serialport.rubyforge.org/README

Alien developed the Alien RFID Custom Application (ARCA) kit to accommodate the expanding list of applications for Gen 2 RFID. Alien partners and end users like you are discovering many new ways to use RFID technology and need the ability to adapt off-the-shelf readers to new applications. ARCA enables you to deliver more effective solutions, more quickly for emerging applications. Thanks for using Alien Technology RFID products. For support please contact support@alientechnology.com.

The Alien RFID Team