Legal Notices
Copyright © 2007 Alien Technology Corporation. All rights reserved.

Alien Technology Corporation has intellectual property rights relating to technology embodied in the products described in this document, including without limitation certain patents or patent pending applications in the U.S. or other countries.

This document and the products to which it pertains are distributed under licenses restricting their use, copying, distribution and decompilation. No part of this product documentation may be reproduced in any form or by any means without the prior written consent of Alien Technology Corporation and its licensors, if any. Third party software is copyrighted and licensed from Licensors. Alien, Alien Technology, the Alien logo, Nanoblock, Fluidic Self Assembly, FSA, Gen2Ready, Squiggle, RFID Gateway, Nanoscan and other graphics, logos, and service names used in this document are trademarks of Alien Technology Corporation in the U.S. and other countries. All other trademarks are the property of their respective owners. U.S. Government approval required when exporting the product described in this documentation.

Federal Acquisitions: Commercial Software -- Government Users Subject to Standard License Terms and Conditions. U.S. Government: If this Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in the Software and accompanying documentation shall be only as set forth in this license; this is in accordance with 48 C.F.R. 227.7201 through 227.7202-4 (for Department of Defense (DoD) acquisitions) and with 48 C.F.R. 2.101 and 12.212 (for non-DoD acquisitions).

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE HEREBY DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.
# Table of Contents

## CHAPTER 1 INTRODUCTION

- Audience ......................................................................................................................... 1
- Overview .......................................................................................................................... 1
- Installation ....................................................................................................................... 3

## CHAPTER 2 SUPPORTING DATA TYPES

- Introduction ................................................................................................................... 4
- The ReaderInfo Class ................................................................................................. 5
- The TagInfo Class ......................................................................................................... 6
- The NotifyInfo Class ................................................................................................. 7
- The AlienIOInfo Class ............................................................................................... 8
- The UpgradeInfo Class ............................................................................................. 9

## CHAPTER 3 THE CLSREADERMONITOR CLASS

- Introduction .................................................................................................................. 10
- Serial Discovery and Monitoring ................................................................................ 11
- Network Discovery and Monitoring .......................................................................... 12

## CHAPTER 4 THE CLSREADER CLASS

- Introduction .................................................................................................................. 13
- Instancing a Reader from the clsReaderMonitor Discovery Class ......................... 13
- Instancing a Reader Directly On a Serial Connection ............................................... 13
- Instancing a Reader Directly on the Network ............................................................ 14
- Opening and Closing Connection to a Reader ............................................................ 14
- Communication with a Reader .................................................................................... 16

## CHAPTER 5 TAG WORKS

- Introduction .................................................................................................................. 17
- Programming Tags ....................................................................................................... 17
- Reading Tags ................................................................................................................. 20
- Parsing TagList Data ................................................................................................... 21

## CHAPTER 6 THE CALIENSERVER CLASS

- Introduction .................................................................................................................. 22
- Asynchronous Notifications ...................................................................................... 23
- Retrieve Notifications Synchronously ...................................................................... 23
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Class Diagram of the Alien .NET Library</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Alien RFID Library enumerations</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>ReaderInfo class</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Class TagInfo</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Class NotifyInfo</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Class AlienIOInfo</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>Class UpgradeInfo</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>Sample source code demonstrating Alien Discovery and Monitoring</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Creating an instance of the reader from the Reader Monitoring</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>Initializing reader for serial communication</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>Initializing reader for network communication</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Logging to the network</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>Programming Tags with the “Programming Tags”</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>Obtaining reader internal Tag list</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>Obtaining and parsing TagList using the Ex6 (Tag Lists) sample application</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Parsing string response to the TagList command into array of TagInfo data</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>Members of the CAlienServer class</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>Ex7 (Network Listener) example application in actions</td>
<td>23</td>
</tr>
</tbody>
</table>
CHAPTER 1
Introduction

The Visual Basic Developers Guide provides basic instructions for programmatically controlling an Alien® reader using the Microsoft Visual Basic 6.0 Software Development Environment and the AlienRFID1.dll class library with its COM interface supplied by Alien Technology® as a part of the Developer's Kit.

Audience

For the purposes of this document, we assume the readers of the Visual Basic Developers Guide:

* are competent PC users
* have minimal previous knowledge of radio-frequency identification technology
* are experienced in Visual Basic software development.

Overview

A full featured Alien RFID Reader can be programmatically controlled using a number of systems and languages. This document focuses on controlling the reader using the AlienRFID1.dll with its exposed COM interface supplied with the Visual Basic (6.0) developers’ kit.

Though the AlienRFID1.dll has been developed and built in the Microsoft .NET Framework, its COM interface intends for use with other development environments like, for example Visual Basic 6.

In this document terms Alien .NET Library (Library) and Alien .NET API have been used interchangeably and all reference the AlienRFID1.dll.

Alien Technology provides a number of working sample applications with their source code developed in Visual Basic 6.0 as Alien VB6 SDK demonstrating how to use COM interface of the AlienRFID1.dll described in the AlienRFID1.tlb type-library.

The class library contained within AlienRFID1.dll provides type structures and classes that constitute discrete functional groups for controlling various aspects of the reader:

- **Storage Types** – Data types for handling data about readers and RFID tags. Other classes use these types to pass to functions and user applications information about the state of readers connected to the system.
- **Discovery and Monitoring features** – Classes for discovering the location of readers connected via serial ports or networks as well as for monitoring tags’ status and reader’s notifications.

- **Utility features** – Usually static functions that are used for example for stateless data conversion.

The Figure 1 illustrates overall structure of the Alien .NET API.

![Class Diagram of the Alien .NET Library.](image)

**Figure 1**: Class Diagram of the Alien .NET Library.

Interfaces in the left part of the diagram represent features available to COM users.
Installation

To use the Alien RFID libraries from a Visual Basic application the Alien .NET Library and its COM interface must be registered on the development system using the Setup program provided on the development CD.

NOTE: Please uninstall any previous versions of the Alien .NET Library using the Control Panel → Add or Remove Programs. If you choose to delete Alien .NET Library files manually, make sure you un-register them prior to deleting.

You may want to remove manually any additional copies of the AlienRFID1.tlb and supporting files that have been left after uninstalling older versions of the Alien .NET Library.

Please use the Alien setup package that includes the following items:

- **Setup.exe, settings.ini, and dotnetfx.exe** – Provide checking for presence of the MS .NET Framework v.1.1 and install it if needed. Than setup program will automatically proceed with installation of the AlienRFID1setup.msi. You must start with the setup.exe if you didn’t have .NET Framework (v.1.1) installed on your computer.

- **AlienRFID1setup.msi** – Installs the Alien .NET Library and registers its COM interface in the Windows Registry. The AlienRFID1.tlb (type library for referencing from VB6 IDE) goes to the <WindowsFolder.>

- **AlienRFID1mm.msm** – Merge Module for redistribution of the Alien RFID Library. When included into a developer’s installation package, it provides installation of the AlienRFID1.dll with its satellite files and registration of the COM interface in the Windows Registry of the end-user computer. It does NOT install the Microsoft .NET Framework v.1.1. So the latter must be present on the end-user system or installed with the developer’s setup package before running the merge module

- **The Alien VB6 example projects with their source code** should be just copied to the developers’ desired directory.

- **The Alien .NET API Documentation.chm** file provides the MSDN-style help for all classes, methods, properties, and types of the Library and can be copied to any desired location.

To use functions provided by the Library, a reference to the AlienRFID1.tlb must be included in VB6 project. From the Visual Basic 6.0 IDE use the “Project/References” menu and check the entry appearing for the “Alien API for RFID.”
CHAPTER 2
Supporting Data Types

Introduction

An Object Browser of Visual Studio .NET or Visual Basic IDE allows one to examine data types and interfaces within the Alien .NET API in more details.

There are several supporting classes for working with reader and tag data: a number of enumerations specifying some frequently used constants, a class with static utility methods, and classes for storing information about reader, tag, tag notification, IO event, reader upgrade etc.

The following figure details members of Alien .NET enumerations:

![Alien RFID Library enumerations](image)

Figure 2: Alien RFID Library enumerations

Each of the following storage classes has a copy constructor and a corresponding COM interface exposed by Library. Also, there are related utilities in the AlienUtils class to parse a string from reader (in Text or XML format when applicable) and return a data object.
The ReaderInfo Class

Whether found through a call to the clsReaderMonitor.CheckComPorts() method or by a multicast “Heartbeat” (see below), host applications are notified of available readers via an event passing up an instance of the ReaderInfo class. It contains key information that allows a software system to identify and contact a reader. Information such as the reader name, type and address is provided in this type.
The TagInfo Class

Figure 4: Class TagInfo

TagInfo is a type for holding information about tags. This type allows one to track the Tag ID, the CRC for the Tag ID, the date and time the tag was last observed by the reader as well as number of times tag has been observed. Functions such as AlienUtils.ParseTaglist() return arrays of TagInfo objects from raw string data read by the reader.
### The NotifyInfo Class

NotifyInfo is a storage type for information sent by the reader automatically as a Notification message and contains corresponding fields and properties. Client application can use Alien Utilities methods that parse an incoming notification string either in the Text format or as XML and return an instance of class NotifyInfo.

#### NotifyInfo Members

<table>
<thead>
<tr>
<th>Public Instance Constructors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NotifyInfo</td>
<td>Overloaded. Initializes a new instance of the NotifyInfo class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Instance Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CommandPort</td>
<td>Gets / Sets string representing current network port number. Default = “23”.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>Sets / Gets string with current IP Address of a reader.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>Represents MAC Address of a reader.</td>
</tr>
<tr>
<td>ReaderName</td>
<td>Sets / Gets string with Reader Name.</td>
</tr>
<tr>
<td>ReaderType</td>
<td>Sets / Gets string with reader type.</td>
</tr>
<tr>
<td>Reason</td>
<td>Sets / Gets string representing reason of notification.</td>
</tr>
<tr>
<td>StartTriggerLines</td>
<td>Sets / Gets string representing external digital input line that started the reader’s automode sequence.</td>
</tr>
<tr>
<td>StopTriggerLines</td>
<td>Sets / Gets string representing external digital input line that stopped the reader’s automode sequence.</td>
</tr>
<tr>
<td>TagList</td>
<td>Sets / Gets array of type TagInfo representing Alien TagList.</td>
</tr>
<tr>
<td>Time</td>
<td>Sets / Gets string representing time of Notification.</td>
</tr>
</tbody>
</table>
The AlienI0Info Class

AlienI0Info class contains properties characterizing an asynchronous Digital Input or Output event as a part of Alien IO Stream as well as members of an IOList returned by the reader synchronously.

The static AlienUtils.ParseAlienIOEvent() method returns an instance of this class after parsing a string received from the reader.

NOTE: Older readers don’t support IO Stream and IOList.
The UpgradeInfo Class

Represents information included as a parameter in the UpgradeProgress and UpgradeComplete events after calling the UpgradeFirmware() method.

This class also overrides the ToString() method to return a multi-line string with a list of current properties formatted as <name>: <value> pairs.

<table>
<thead>
<tr>
<th>UpgradeInfo Members</th>
<th>UpgradeInfo Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Instance Constructors</td>
<td>Overloaded. Initializes a new instance of the UpgradeInfo class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Public Instance Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CanCancel</code></td>
<td>Provides true/false information about if it is safe to interrupt with running upgrading process and cancel it.</td>
</tr>
<tr>
<td><code>Cancelled</code></td>
<td>Provides true/false information if upgrade process has been cancelled.</td>
</tr>
<tr>
<td><code>Completed</code></td>
<td>Provides true/false information about completion of the upgrade process.</td>
</tr>
<tr>
<td><code>Message</code></td>
<td>Provides description about current upgrading tasks and status.</td>
</tr>
<tr>
<td><code>PercentDone</code></td>
<td>Provides percentage status about current upgrading task. (Not available for all upgrading tasks.)</td>
</tr>
<tr>
<td><code>Reconnected</code></td>
<td>Provides true/false information about status of restoring connection after upgrade.</td>
</tr>
<tr>
<td><code>StateRestored</code></td>
<td>Provides true/false information about status of restoring reader's state after upgrade.</td>
</tr>
<tr>
<td><code>UpgradeResult</code></td>
<td>Provides information about status of firmware upgrading with an uploaded file. Value of 0 indicates that reader has been upgraded successfully; value of -1 indicates that upgrade process has been interrupted by a failure and/or cancelled; other error codes possible.</td>
</tr>
<tr>
<td><code>UploadResult</code></td>
<td>Provides information about status of uploading a firmware file to reader. Value of 0 indicates that firmware file has been uploaded successfully; value of -1 indicates that upload process has been interrupted by a failure and/or cancelled; other error codes possible.</td>
</tr>
</tbody>
</table>

Figure 7: Class UpgradeInfo
CHAPTER 3
The clsReaderMonitor Class

Introduction

In order to use and control an Alien reader it first has to be discovered. This can mean discovery on the network or discovery on serial ports. The clsReaderMonitor contained within the Alien RFID Library is a class that can automatically search for and discover readers using each of these connection modes. Use methods StartListening() and StopListening() for starting and stopping monitoring over both TCP and Serial connections.

All discovered readers information is stored in two separate collections and monitored separately depending on the values of boolean ComPortsMonitoring and NetworkMonitoring properties. The methods GetReaderList() and ClearAllReaders() apply to all readers on both serial and network connections.

Similarly, the events ReaderAdded, ReaderRenewed, ReaderRemoved, and ReaderListUpdated get raised for both serial and network connected readers and contain information about all discovered readers.

In order to receive clsReaderMonitor events client application must subscribe to them and implement event handler procedures.

In Visual Basic this is done by declaring an object of clsReaderMonitor WithEvents and writing event-handler routines.
Serial Discovery and Monitoring

If set so, the clsReaderMonitor checks all available com ports (excluding modem ports) listed in the registry key HKLM\HARDWARE\DEVICEMAP\SERIALCOMM\ and the sub-key Device.

Discovery of Alien readers attached to serial ports of a host computer supports manual calls to the CheckComPorts() and GetReaderListOnSerial() methods. Both methods return synchronously after completion of operation. The CheckComPorts() method returns with an empty string or “No reader found”.

NOTE: If a serial connection to the reader has already been open manually or using another software component, the clsReaderMonitor will not be able to discover this reader. And opposite, if serial discovery has been enabled, this can preclude another software component or object (such as the clsReader) from connecting to the discovered reader.
Network Discovery and Monitoring

Each Alien reader is configured by default to send out broadcast messages to its local subnet. These messages are small XML documents detailing the reader type, name, and contact information. By listening for these messages, instances of the clsReaderMonitor class can identify and report back details of readers that are alive and on the network. To enable network monitoring set the NetworkMonitoring property to true.

The class instance will catch reader “heartbeats”, decode them into ReaderInfo type objects, and update its internal Reader list with this information, raising either ReaderAddedOnNetwork, or ReaderRenewedOnNetwork events to an application.

Part of the heartbeat sent out by the reader indicates the time until the next heartbeat is expected. If this time expires before a new heartbeat is received, then the class will assume the reader has gone offline and will raise the ReaderRemovedOnNetwork event.

At any time the current list of the on-line networked readers can be obtained by calling the clsReaderMonitor.GetReaderListOnNetwork() method. This will return a number of currently on-line readers and an array of ReaderInfo objects the same way as illustrated above.

The method ClearNetworkReaders() erases all items from the current list of readers discovered on network. It doesn’t affect list of serial-connected readers. Therefore, if there are readers on Com ports and ComPortsMonitoring property is set to true, method GetReaderList() can return not empty.

Please see examples above and SDK for coding guidelines
CHAPTER 4  
The clsReader Class

Introduction

The clsReader class is used for communicating with a reader either over the network or serial port. Typically the clsReader object will be initialized with data obtained from a clsReaderMonitor class, discussed in the previous section. However if the location (either serial port name or network address) is known, a clsReader object can be instantiated directly without the need of any discovery class.

Once a valid reader object is available, it offers the user a number of simple commands that implement the full command set described in the Alien Reader Interface Guide.

Instancing a Reader from the clsReaderMonitor Discovery Class

If a discovery class is used (see previous section), any readers that are found on the network or serial ports will result in a ReaderInfo data type object being passed to the application. To convert this data into a clsReader object, declare an instance of the clsReader class and use the clsReader.ReaderSettings property:

```vba
Private Sub mMonitor_ReaderAdded(ByVal r As ReaderInfo)  'Marco Monza
Dim reader As New clsReader
reader.ReaderSettings = r  'OR
reader.WriteSettings r
End Sub
```

Figure 9: Creating an instance of the reader from the Reader Monitoring.

Instancing a Reader Directly On a Serial Connection

If it is known that a reader exists on a specified serial port, a new Reader object can be initialized with default or pre-defined settings directly without having to use discovery.
Private Sub btnConnect_Click()
    Dim stemp As String
    mReader.InitOnCom(1) ‘Initialize reader object on COM1
    stemp = mReader.Connect()
    textBox1.Text = textBox1.Text & vbCrLf & stemp
End Sub

Figure 10: Initializing reader for serial communication

In the example above, a new reader object is created and initialized on Com port using InitOnCom() function.

This is all that is required to instance a new reader object. This will tell the mReader object that it has to prepare to serial communication with reader. Using properties of clsReader, you can specify other Com port settings prior to opening port.

Instancing a Reader Directly on the Network

If it is known that a reader exists at a specified network address, a new reader object can be created directly without having to use the discovery class in a manner similar to instancing a reader directly on a serial port.

In this case function InitOnNetwork() should be used passing two argument required for initializing an object for network communication: a string for IPAddress and an integer for port number.

mReader.InitOnNetwork "10.10.1.111", 23

Figure 11: Initializing reader for network communication

NOTE: If the InitOnCom() or InitOnNetwork() functions had been called on a connected object of class clsReader, this will cause the existing connection to be closed.

Opening and Closing Connection to a Reader

Once a Reader object has been instanced and its connection settings configured, a connection to it can be opened and the reader can be used. This is achieved using a single Connect() method

Calling this method will open the connection either serial or networking. This method returns synchronously after finishing operation. The return value is a string indicating status:

- “Already connected”, if object had been connected before calling this method.
- “Connected”, on success.
“Can’t connect” or “Alien caught exception” with an exception message appended in case of a failure.

In case of serial connection, library opens com port and sends to the readers a set of basic commands in order to verify if the reader can respond.

On success, this method raises **Connected** event to the calling application. This event can be raised on a separate thread depending on the connection type.

Use boolean **IsConnected** property to test if connection is open.

In case of network interface type, although connected, at this point the application cannot yet make use of the reader object. For that, a second method must be issued to login to the reader as shown in the code sample:

```vbnet
If meReaderInterface = ComInterface.enumTCPIP Then
    lblStatus.Caption = "Logging in..."
    If Not mReader.Login("alien", "password") Then
        lblStatus.Text = "Login failed! Calling Disconnect()..."
        mReader.Disconnect()
    Else
        lblStatus.Text = "Logged in - OK!"
    End If
End If
```

**Figure 12: Logging to the network**

All network based readers require a username and password to use them. By default all network readers will use "alien" as the username and "password" as the password. Once connected and logged in, these can be changed and verified using the **clsReader.Password** and **clsReader.UserName** properties.

Failure to set the correct username and password when logging in will return a boolean value of “false” from a call to the above function.

Finally, a connection to a reader can be closed using the **Disconnect** method. This method returns synchronously after closing connection and destroying all supporting threads.

The return value is:

- Empty string on success
- "Not connected" if not applicable
- “Alien caught exception: " with exception message appended in case of any unexpected failure.

On success, this method raises **Disconnected** event to the calling application with “Disconnected by client” string argument value.
Communication with a Reader

All commands to and from the reader are ASCII text based messages that take the form of command-response pairs. The `clsReader` class provides two generic methods called `SendReceive()` and `Send()` for ASCII based communication with the reader.

All of the reader commands and their expected responses are discussed in the Alien Reader Interface Guide. However these methods requires detailed knowledge of the reader command set and requires any results to be parsed and processed.

The `SendReceive()` is synchronous method. It takes an input string containing a reader command with required parameters and a boolean flag indicating whether the reader response should include user prompt for further operations. Upon completion of operation, it returns a string with reader response parsed for convenient use. In case of failure an exception will be thrown.

The `Send()` is an asynchronous void method. It takes the first parameter same as `SendReceive` method. The second parameter is a boolean flag indicating whether the first parameter should be considered as “raw data” or not. It returns immediately after sending data to the reader. In case if connection was not established or has been lost, the `Disconnected` event shall be raised. Any exception happened on the caller’s thread shall be re-thrown to the caller. The reader’s response should be taken in one of the available events raised asynchronously and on separate threads:

- **DataReceived** event containing a part of reader’s response usually terminated by the “carriage return” and “new line” characters.
- **MessageReceived** event containing complete reader response terminated by null-character.

However these methods require detailed knowledge of the reader command set and requires any results to be parsed and processed. To make life simpler for the developer, the reader object also supports many additional methods that directly correspond to the reader command set. For example, the reader object has a property called `clsReader.PersistTime`. This property returns an integer number. It is effectively the same as calling the `clsReader.SendReceive("get PersistTime", false)` method and then parsing the string reply into an integer.
CHAPTER 5
Tag Works

Introduction

The reader can program an individual tag and/or read tags’ data.
While older readers supported only working with EPC ID field of the EPCglobal Class1 tags, the ALR-9800 readers utilizing Class1 Generation2 tags can read and program other memory banks also. There are many reader’s properties that affect acquiring or writing data from/to tags. Please refer to the Alien Reader Interface documents for more details.

Programming Tags

Alien reader can be used for programming data into RFID tags’ EPCglobal ID field and (when supported by reader firmware version and tags) other EPC Class1 Gen2 memory banks.

The following programming related methods and properties are available with the Alien .NET API on an instance of the clsReader object:

- **Commands supported by all readers**
  - ProgramTag
  - EraseTag
  - LockTag
  - KillTag
  - ProgIncrementOnFail
  - ProgramPassCode
  - ProgramID
  - ProgAttempts
  - ProgEraseAttempts
  - ProgReadAttempts
  - ProgSucessFormat

- **Commands supported by ALR-9800 (subject to firmware version):**
  - PrgramEPC
  - ProgramAndLockEPC
  - ProgramUser
  - ProgramAndLockUser
  - ProgramKillPwd
The sample application “ProjectProgrammingTags” (Figure 13) included into the Alien Visual Basic 6 SDK with its source code provides details on various programming features of Alien readers and the Alien .NET API.
Figure 13: Programming Tags sample application
Reading Tags

The most common operation to perform with a reader is to read tags. This is done in a number of ways depending on the read environment and requirements.

The simplest way to read tags is to use the `TagList` property:

```csharp
string result = mReader.TagList;
```

Figure 14: Obtaining reader internal Tag list

The format of the reply depends upon the value of the reader's `TagListFormat` property. Valid options are "Text" or "XML."

Please see next section for details on working with TagList.

Figure 15: Obtaining and parsing TagList using the Ex6 (Tag Lists) sample application
Parsing TagList Data

The reader can return lists of tags in several formats: Standard Text, XML and abbreviated. The AlienUtils class provides static methods and the clsReader non-static methods for parsing this tag list string into an array of TagInfo objects (described above) that may be more convenient for developers to use.

Figure 16: Parsing string response to the TagList command into array of TagInfo data.
CHAPTER 6
The CAlienServer Class

Introduction

The CAlienServer class provides methods for listening asynchronous messages as Alien Notifications and/or Tag- and IO-Stream events sent by Alien Readers over network.

You can create several server objects for listening on different types of messages and/or on messages from different readers using different port numbers. If you don’t specify explicitly, the first available IP Address from the list resolved for this host machine will be used to listen on.

When you create an instance of the CAlienServer class, there is a TCP socket created for listening to incoming connections. You can turn listening ON and OFF by calling methods StartListening() and StopListening().

The following figure lists members of the CAlienServer class:

Figure 17: Members of the CAlienServer class
Asynchronous Notifications

**CAlienServer** maintains a collection of established connections identifying every connection with unique identifier (GUID.)

User can subscribe to the events raised by the server to receive updated information about established / lost connections as well as messages sent by readers in AutoMode and when Notification mode is set ON and configured correspondingly.

Please note that not all readers implement streaming modes. Particularly, older readers like ALR-9780 will support only Tag Notifications.

Retrieve Notifications Synchronously

Though, it could be difficult to deal with multithreading in a single-thread oriented development environment like Visual Basic 6. So, the Library contains synchronous methods for collecting Alien Notifications.

In case of NO subscribers to the ServerMessageReceived event, all incoming messages get collected in an internal queue limited to a maximum defined by the MaxQueuedMessages property (default is 100.) A client application can retrieve these messages synchronously by calling methods GetCurrentNotifications() and GetCurrentIOEvents(). Returned messages will be cleared from the queue by these calls.

This feature intends specifically for Visual Basic 6 development and is demonstrated in the Project #7 – Ex7(Network Listener) as shown below:

Figure 18: Ex7 (Network Listener) example application in actions.