SDD

Wayne.ForecourtControl.OptBridge

Table of Contents

1 Document information

File: SDD_Wayne.ForecourtControl.doc

1.1 Revision history

Revision	Author	Date	Change description
2.0	Roger Månsson	2006-07-25	Created
2.1	Roger Månsson	2007-01-09	Added COM interfaces, tanks and some more descriptions.

1.2 Purpose and scope

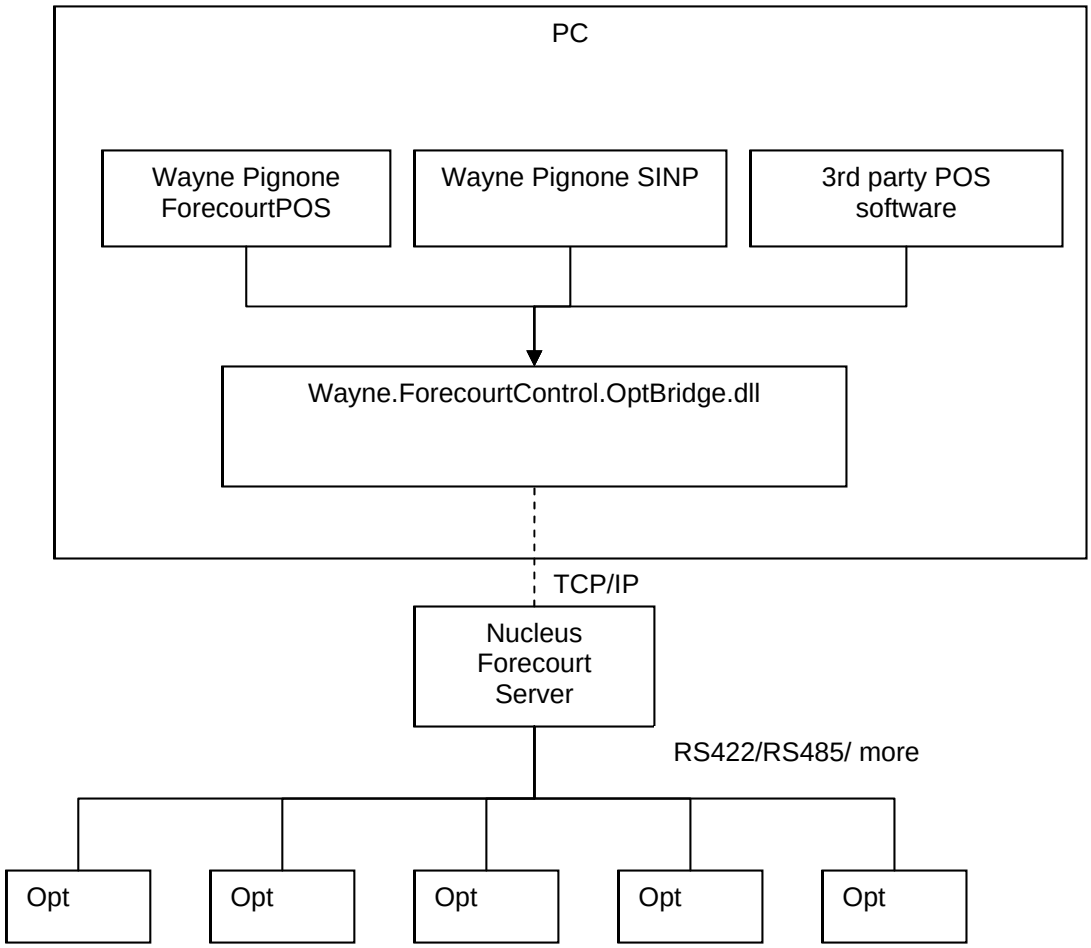
1.3 Abbreviations and acronyms

Abbreviation	Meaning

1.4 References

2 Overview

The forecourt Opt bridge control library enables applications to communicate with terminals that are physically connected to the forecourt links tunneling the data through the forecourt server.



3 Forecourt Opt bridge

The OptBridge object is the main object in the library.

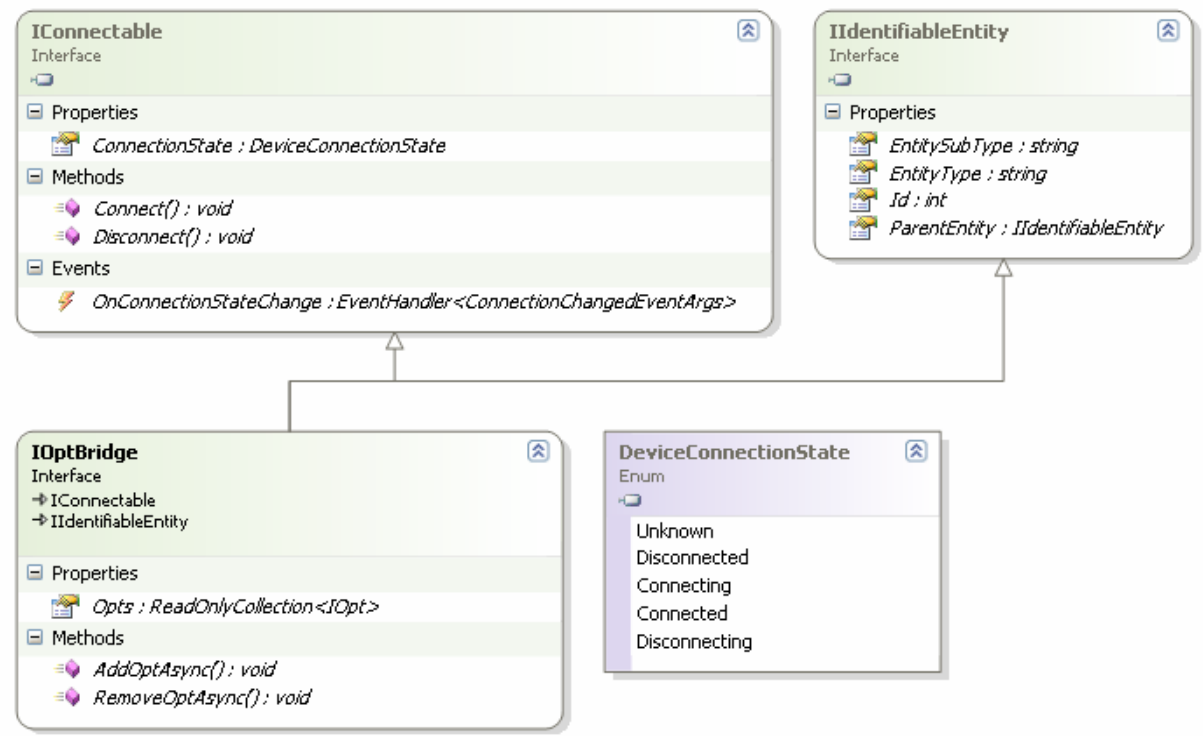
After creating the OptBridge object, it is still not initialized, and has the ConnectionState=*Disconnected*. In order to perform any operations, the OptBridge must be connected to the forecourt server. This is performed using the *Connect*() method using a connection string. When connecting to the forecourt server, the connection string is a comma-separated string with the connection parameters.

Example:

```
ClientId=210,ClientName=TestApp,Host=192.168.1.1
```

Parameter	Range	Description
ClientId	1-65,535	A unique client number. 0 is not allowed, and 200-205 is reserved for internal use in the Nucleus application.
Client name	---	A string representing the application name. This is only used for debug purposes and can be left out.
Host		IP Address or DNS name of the forecourt server to connect to.

When connect has been called, the connection state will change to *Connecting*. When the forecourt control is finally connected and completely initialized, the connection state will change again to *Connected.* Connection state changes are notified using the OnConnectionStateChanged event.



3.1 Managing the Opt bridge connection

Create an instance of the IOptBridge object. This object is uninitialized, and requires to be connected in order to perform any operations.

3.1.1 Create the forecourt control object

The syntax and procedure is highly dependant on the implementation platform. This example only shows how it would have been written in C#.

Application		ForecourtOptBridge
IOptBridge ob = new OptBridgeClass()	→ ←	

3.1.2 Connect to the forecourt server

Application		ForecourtOptBridge
Call the connect method with a connection string.		
fc.Connect(string connectionString)	→	
Event signaling that the forecourt control object tries to connect to the Forecourt server		
	←	**OnConnectionStateChange** -Connecting
Event signaling that OptBridge object has succeeded in connecting to the forecourt server and that all initializations are done.		
	←	**OnConnectionStateChange** - Connected

The connection state will first change to 'Connecting', then to Connected. It is only when the connection state is 'Connected' that operations can be performed in the interface.

Example on a connection string.

string connectionString =
"Host=192.168.1.1,Port=11028,ClientId=777,ClientName=Test,LogFile=OptBridge"

3.1.3 Connection link lost or restored during operation

Application		ForecourtOptBridge
Connection to the forecourt server is lost.		
	←	**OnConnectionStateChange** - Connecting
The OptBridge object will automatically try to reconnect to the forecourt server.		
	←	**OnConnectionStateChange** - Connected.

When the connection state not is 'Connected', all objects that are hieratically referenced from the OptBridge are obsolete. When the connection is restored, the objects can be located again from the arrays in the OptBridge object.

3.1.4 Closing the interface

Application		ForecourtOptBridge
Application calls Disconnect		
fc.Disconnect()	→	
OptBridge signals that it has initiated the disconnection		
	←	**OnConnectionStateChange** Disconnecting
OptBridge signals that it has completed the disconnection.		
	←	**OnConnectionStateChange** Disconnected

3.2 Adding and removing terminals

3.2.1 Adding a terminal

Application wants to add terminal 3.

Application		ForecourtOptBridge
Application calls AddOptAsync for terminal 3		
ob.AddOptAsync(3)	→	
Response that signals that the command was completed.		
	←	**OnAddOptAsyncCompleted**
Now, if the add was successful, the terminal should be available in the OptBridge.Opts collection. Note that we have to search the collection for the terminal, since we can only get the IOpt object by the collection index.		
IOpt opt= null **for(int i =0; i< ob.Ops.Count; i++)** **{** **if(ob.Opts[i].Id==3)** **opt = ob.Opts[i];** **}**	→	

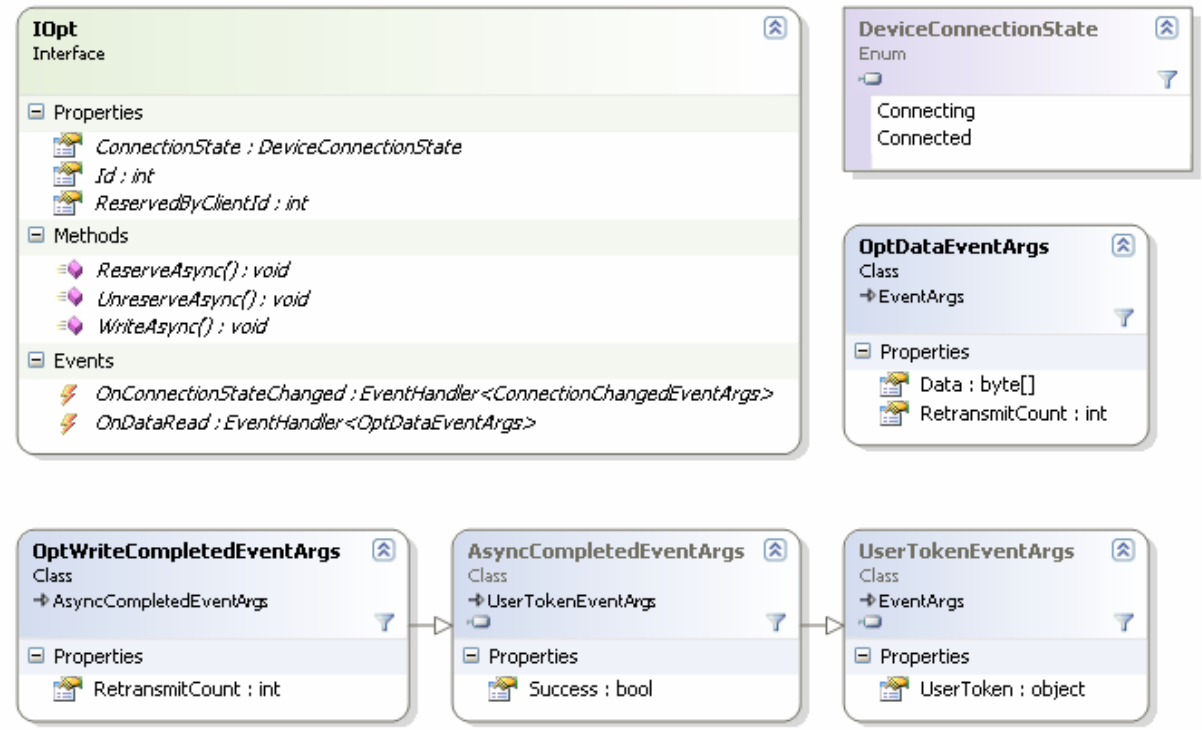
3.2.2 Removing a terminal

Application wants to remove terminal 3.

Application		ForecourtOptBridge
Application calls RemoveOpt for terminal 3		
ob.RemoveOptAsync(3)	→	
Response that signals that the command was completed.		
	←	**OnRemoveOptAsyncCompleted**
Now, if the remove was successful, the terminal should be removed from the IOptBridge.Opts collection.		

4 Outdoor payment terminals (Opt)

One Opt object represents a communication point to one physical terminal. In order to communicate with the terminal the application must reserve it first.



4.1 DART Protocol TerminalData example:

The TerminalData expected to contain the full DART data frame, the Forecourt server will only append Protocl releated information, se below.

Transmitted DART data frame

ADR	CTRL	**Data in Write(TerminalData)**	CRC-1	CRC-2	ETX	SF

Data in *grey* is added by the forecourt server.

Terminal polling and Protocol level ACK handling is made by the forecourt server.

4.2 Reserving the Opt

4.2.1 Reserving the terminal

Application		ForecourtOptBridge
Application obtains a reference to the first terminal in the Opts list.		
IOpt opt = ob.Opts[0]	→ ←	
Application calls ReserveAsync		
ob.ReserveAsync()	→	
Response that signals that the command was completed.		
	←	**OnReserveCompleted**

4.2.2 Unreserving the terminal.

Application		ForecourtOptBridge
Application unreserves the terminal.		
Opt.UnreserveAsync(3)	→	
Response that signals that the command was completed.		
	←	**OnUnreserveCompleted**

4.3 Communicating

The terminal is reserved and ready and connected, so now the application can communicate with the terminal.

4.3.1 Sending data to the terminal

Application		ForecourtOptBridge
Application obtains a reference to the first terminal in the Opts list.		
IOpt opt = ob.Opts[0]	→ ←	
Application calls WriteAsync		
opt.WriteAsync()	→	
When the write has either failed or the write has been successfully sent to the terminal, the response event is raised.		
	←	**OnWriteAsyncCompleted**

4.3.2 Receiving data from the terminal

Application		ForecourtOptBridge
Terminal sends data that is received by the application in the OnDataRead event.		
	←	**OnDataRead**

5 Namespace Wayne.ForecourtControl.OptBridge

Interfaces

IOpt	The OPT object is used to communicate with an outdoor payment terminal connected to the forecourt server. Only one client can control the interface, and the interface must be reserverd by the client before the write() method is accepted. OnDataRead is also only signalled to the client that owns the reservation.
IOptBridge	Root object for the Opt communication using a application-layer bridge. The interface provides functionality to get access to single Opts, and change the configuration.

Classes

OptDataEventArgs	Event argument for a data read from the Opt
OptWriteCompletedEventArgs	The EventArgs is used when a Opt write has completed.

5.1 Interfaces

5.1.1 Interface IOpt

```
public interface IOpt
```

Summary
The OPT object is used to communicate with an outdoor payment terminal connected to the forecourt server. Only one client can control the interface, and the interface must be reserverd by the client before the write() method is accepted. OnDataRead is also only signalled to the client that owns the reservation.

Properties

ConnectionState `Lib.DeviceConnectionState`	R	Current state of the connection to the terminal. The Connection state will only be either Connected when the terminal is online or Connecting if the terminal is not responding.
Id `int`	R	Logical terminal number / filling position.
ReservedByClientId `int`	R	If the terminal is reserved, the clientId of the reserving IOptBridge client will be reported here. If the terminal not is reserved, it will be 0.

Methods

ReserveAsync
```
public void ReserveAsync(EventHandler{Wayne.Lib.AsyncCompletedEventArgs}
reserveCompleted, object userToken);
```
Reserves this device for exclusve control from this instance.

reserveCompleted	Callback delegate that will be invoked on completion.
userToken	User token object that will be returned in the completion callback

UnreserveAsync
```
public void UnreserveAsync(EventHandler{Wayne.Lib.AsyncCompletedEventArgs}
unreserveCompleted, object userToken);
```
Cancel device reservation. This command is only allowed after a successful call to ReserveAsync() by

the same client.	
unreserveCompleted	Callback delegate that will be invoked on completion.
userToken	User token object that will be returned in the completion callback

WriteAsync
```
public void WriteAsync(Byte[] terminalData, bool waitForSendOk,
EventHandler{Wayne.ForecourtControl.OptBridge.OptWriteCompletedEventArgs}
writeCompleted, object userToken);
```
The data is transparently sent to the terminal, only appending protocol specific information. If waitForSendOk=false, the request will complete as soon as the data has been sent to the forecourt server. If it is true, the request will be complete after the data is actually sent to the terminal.

terminalData	
waitForSendOk	If this flag is set, the writeCompleted will not be invoked until the data is actually sent to the terminal.
writeCompleted	
userToken	

Events

OnConnectionStateChange
```
public EventHandler{Wayne.Lib.ConnectionChangedEventArgs}
OnConnectionStateChange;
```
Event signalling that the connection state of the terminal has changed.

OnDataRead
```
public EventHandler{Wayne.ForecourtControl.OptBridge.OptDataEventArgs}
OnDataRead;
```
Fired when data has been read from the terminal. Only the client that has reserved the terminal will receive this event.

5.1.2 Interface IOptBridge

```
public interface IOptBridge
```

Summary
Root object for the Opt communication using a application-layer bridge. The interface provides functionality to get access to single Opts, and change the configuration.

Properties

ClientId `int`	R	The Client id that was specified in the connection string when connecting.
ClientName `string`	R	The Client name that was specified in the connection string when connecting.
Opts `Collections.ObjectModel.ReadOnlyCollection{ Wayne.ForecourtControl.OptBridge.IOpt}`	R	Collection of the currently configured Opts.

Methods

AddOptAsync `public void AddOptAsync(int optId, EventHandler{Wayne.Lib.AsyncCompletedEventArgs} requestCompleted, object userToken);` Configure that the specified opt should be handled and polled. If the request completes sucessfully, the new Opt is added to the Opts collection.	
optId	
requestCompleted	
userToken	

RemoveOptAsync `public void RemoveOptAsync(int optId, EventHandler{Wayne.Lib.AsyncCompletedEventArgs} requestCompleted, object userToken);` Remove the specified opt from the list of connected opts. If the request completes successfully, the Opts collection is modified.	
optId	
requestCompleted	
userToken	

Events

OnConfigurationChanged `public EventHandler OnConfigurationChanged;` Event that notifies clients that a Opt has been added or removed.

5.2 Classes

5.2.1 Class OptDataEventArgs

```
public class OptDataEventArgs : EventArgs
```

Summary
Event argument for a data read from the Opt

Properties

Data `Byte[]`	R	Data from the Opt.
RetransmitCount `int`	R	Protocol layer retransmissions required, before the data was received.

Constructors

`public OptDataEventArgs(Byte[] data, int retransmitCount);` Construction.	
data	Data from the Opt.
retransmitCount	Protocol layer retransmissions required, before the data was received.

5.2.2 Class OptWriteCompletedEventArgs

```
public class OptWriteCompletedEventArgs : AsyncCompletedEventArgs
```

Summary
The EventArgs is used when a Opt write has completed.

Properties

RetransmitCount `int`	R	Number of times that the message was retransmitted before it came through or it was cancelled.

Constructors

`public OptWriteCompletedEventArgs(bool success, object userToken, int retransmitCount);` Construction.	
success	
userToken	
retransmitCount	

Methods