ATIOnet .NET SDK Reference

> **About:** This document describes and explains how to consume the .NET SDK for all ATIOnet APIs.

Document Information	
File:	AN-SDK-Reference
Doc. Version:	1.0
Release Date:	10, July 2016
Author:	ATIOnet LLC

Change Log		
Ver.	Date	Change summary
1.0	10/July/2016	Initial version

Contents

Introduction

The ATIOnet SDK (Software development Kit) helps any developer that wants to interact with ATIOnet Platform. If you are integrating an existing software or want to extract data out of ATIOnet, the SDK will make this much easier, taking care of all the HTTP communication, error handling and retry policies.
ATIOnet SDK can be consumed from any .NET language.

Download / Installation

ATIOnet SDK is hosted in www.nuget.org. NuGet is the package manager for the Microsoft development platform including .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.

Operation Types

ATIOnet SDK is able to handle most of the ATIOnet modules (Operation Types). These operations are very specific to the responsibility they have in the platform, like for example *Authorization*, *FMS (Fuel Management Systems)*, *Loyalty*, etc. Each of this Operation Type, has a corresponding module in the SDK. the complete list of *Operations Types* are listed below.

Auth

The Auth Operation Type is the one in charge of interacting with the authorization engine. The authorization engine is the component that receives authorizations requests and decides if the transaction is approved or not. This SDK Operation Type should be use if you are building a POS or a terminal that will be sending authorization requests to ATIOnet.

FMS

The FMS Operation Type is the one in charge of interacting with the FMS module. ATIOnet supports receiving *Inventory* and *Deliveries* transactions through this interface.

Interface

The Interface Operation Type was design to be used by 3rd party software that need to get information from ATIOnet. With this Operation Type you can download transactions (fleet, loyalty, retail, rejected, approved and exceptions), current account movements, current account balances and loyalty current account.

Loyalty

The Loyalty Operation Type is the one in charge of interacting with the Loyalty module. Using this Operation Type you will be able to send Loyalty transactions to ATIOnet among other features.

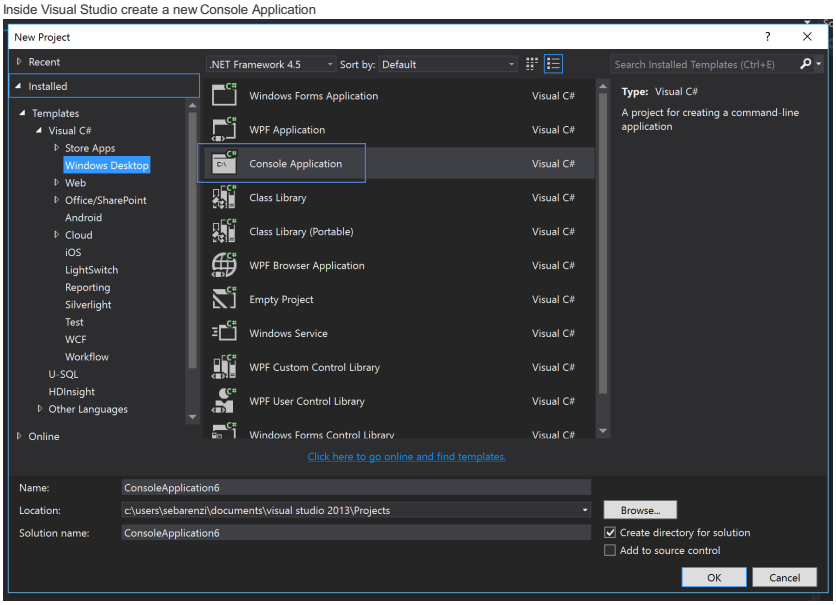
Retail

The Retail Operation Type is the one in charge of interacting with the Retail module. Using this Operation Type you will be able to send Retail information (transactions, batch closes, etc) to ATIOnet among other features.

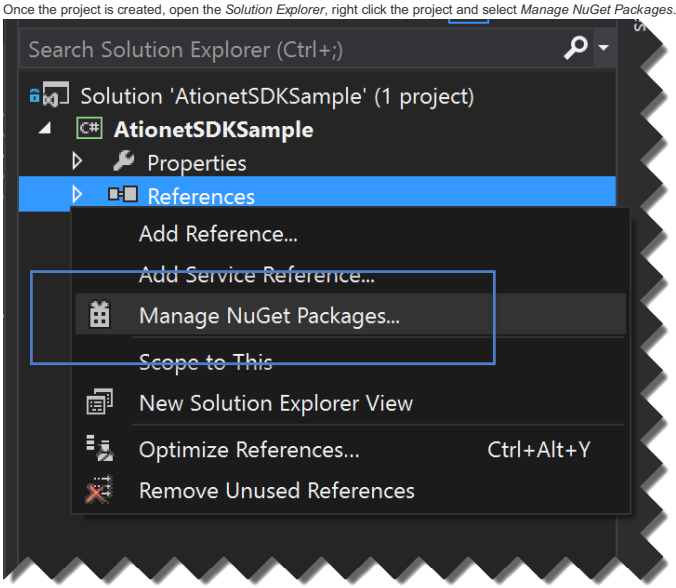
Consuming the SDK

For this example we will use Visual Studio 2013 and we will create a Console Application.

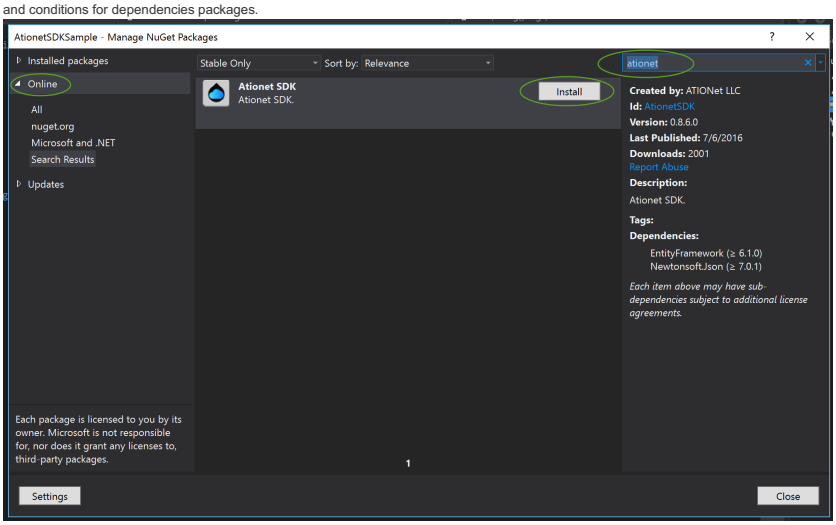
1. Inside Visual Studio create a new Console Application



2. Once the project is created, open the *Solution Explorer*, right click the project and select *Manage NuGet Packages*. This will open a pop up.



3. Inside the NuGet pop up, in the left part select *Online*, then in the right top corner type *ationet*. This will bring the list of packages found with this criteria. In this case only 1, **ATIOnet SDK**. Click *Install*, this will install the ATIOnet SDK packages and dependencies. Please accept terms and conditions for dependencies packages.



4. In the sample below you will see how to download transactions. Open the Program.cs and type the following code:

```
var client = new Ationet.Sdk.Interface.InterfaceOperations("https://native.ationet.com/",
    "[YOUR-USERNAME]", "[YOUR-PASSWORD]");
var transactions = client.GetTransactions("XYZ", "", "", DateTime.Now.AddDays(-1));

foreach (var tran in transactions.Content)
```

```
{
    Console.WriteLine("{0} - {1} - {2}", tran.AuthorizationCode, tran.TerminalCode, tran.ProductAmountDispensed);
}
```

First, you need an instance of the *InterfaceOperations* class. The constructor of this class requires 3 parameters, the url, username and password. Once you have an instance of the *InterfaceOperations* class you can start calling the different methods. Find below the list of all the methods by *Operation Type*:

Operation Types methods

Each Operation Type class has multiple methods to perform specific operation against ATIOnet. Each of this methods has a set of parameters, some of them to determine behaviour and others to filter data.

Auth Methods

Method	Parameters	Description
SendConfirmation	AuthTransactionRequest	Sends a *Confirmation* message to the host (learn more about transactions flow here:
SendConfirmationAsync	AuthTransactionRequest	Async version of the previous method
SendPreAuthorization	AuthTransactionRequest	Sends a *Pre Authorization* message to the host (learn more about transactions flow here:
SendPreAuthorizationAsync	AuthTransactionRequest	Async version of the previous method
SendSale	AuthTransactionRequest	Sends a *Sale* message to the host (learn more about transactions flow here:
SendSaleAsync	AuthTransactionRequest	Async version of the previous method

You can download a fully functional sample code from here:
https://github.com/atioint/ationetsdksamples/tree/master/AtionetAuthSample](https://github.com/atioint/ationetsdksamples/tree/master/AtionetAuthSample

FMS Methods

Method	Parameters	Description
UploadNativeDeliveries	TerminalCode (string), trama (List), SystemModel (string), SystemVersion (string)	loren impsum
UploadNativeDelivery	TerminalCode (string), trama (FMSDeliveryData), SystemModel (string), SystemVersion (string)	loren impsum
UploadNativeInventories	TerminalCode (string), trama (List), SystemModel (string), SystemVersion (string)	loren impsum
UploadNativeInventory	TerminalCode (string), trama (FMSInventoryData), SystemModel (string), SystemVersion (string)	loren impsum

Interface Methods

Method	Parameters	Description
BalanceTransferContractToSubAccount		Transfers money from the contract to a sub account
BalanceTransferSubAccountToContract		Transfers money from the sub account to the contract
BalanceTransferSubAccountToSubAccount		Transfers money between sub accounts
BalanceTransferToContract		Transfers (deposit) money to a contract
BalanceTransferToSubAccount		Transfers (deposit) money to a sub account
BalanceWithdrawFromContract		Removes (Withdraw) money from a contract
BalanceWithdrawFromSubAccount		This method removes (Withdraw) money from a sub account
ContractBalanceDownload		This method downloads a contracts balances list
GetDeliveries		This method downloads a deliveries list
GetExceptions		This method downloads a transaction exceptions list
GetInventories		This method downloads an inventories list
GetMovements		This method downloads current account movements
GetRetailBatchCloses		This method downloads retail batch closes
GetRetailTransactions		This method downloads retail transactions
GetStatements		This method downloads the statements
GetTransactions		This method downloads fleet transactions
GetUncontrolledTransactions		This method downloads uncontrolled transactions
InsertFastTrackOrder		This method inserts fast tracks (trip orders)
SubAccountBalanceDownload		This method downloads the balance of the sub accounts

Loyalty Methods

Method	Parameters	Description
SendLoyaltyAccumulation		loren impsum
SendLoyaltyAccumulationAsync		loren impsum

SendLoyaltyBalanceInquiry		loren impsum
SendLoyaltyBalanceInquiryAsync		loren impsum

Retail Methods

Method	Parameters	Description
SendRetailBatchClose		loren impsum
SendRetailBatchCloseAsync		loren impsum
SendSaleRecordsUpload		loren impsum
SendSaleRecordsUploadAsync		loren impsum

Apendix A - Auth Protocol messages

Value	Description
M	Mandatory
M1	Mandatory. The same as original request
M2	Mandatory. The same as pre-authorization
E	Echo
E1	Echo from original request
O	Optional
C	Conditional
C1	At least one field must be send
C2	Mandatory if product quantity is present
C3	At least one field must be sent. The same as original request

Field Name	Size	Type	Pre-authorization		Completion		Contingency		Offline		Cancellation		Void	
			Request	Response	Request	Response	Request	Response	Request	Response	Request	Response	Request	Response
ApplicationType	3	string	M	E	M	E	M	E	M	E	M	E1	M	E1
ProcessingMode	1	string	M	E	M	E	M	E	M	E	M	E1	M	E1
MessageFormatVersion	3	string	M	E	M	E	M	E	M	E	M	E1	M	E1
TerminalIdentification	Var	string	M	E	M	E	M	E	M	E	M	E1	M	E1
DeviceTypeIdentifier	1	string	M	E	M	E	M	E	M	E	M	E1	M	E1
SystemModel	10	string	M		M		M		M		M		M	
SystemVersion	10	string	M		M		M		M		M		M	
TransactionCode	3	string	100	110	120	130	126	130	125	130	400	410	220	230
AccountType	1	string	M	E	M	E	M	E	M	E	M	E1	M	E1
EntryMethod	1	string	M	E	M	E	M	E	M	E	M	E1	M	E1
ServiceCode	1	string												
PumpNumber	2	string		E		E		E		E		E1		E1
ProductCode	4	string	O	C	C2		C2		C2					

ProductUnitPrice	Var	decimal	O	C	O		O		O					
ProductNetAmount	Var	decimal												
ProductTaxes	Var	Dictionary[string, decimal]	O		0		O		O					
ProductAmount	Var	decimal	O	C	C1		C1		C1					
ProductQuantity	Var	decimal	O	C	C1		C1		C1					
TransactionNetAmount	Var	decimal												
ProductData	Var	List[AtionetProduct]	O		O		O		O					
TransactionAmount	Var	decimal	O	C	C1		C1		C1					
UnitCode	Var	string	O	E	O		O		O					
CurrencyCode	3	string		E										
BatchNumber	Var	int		E										
ShiftNumber	Var	string		E										
TransactionSequenceNumber	Var	int	M	E	M	E		E	M	E	M1	E	M1	E
LocalTransactionDate	8	int	M	E	M	E	M	E	M	E	M	E	M	E
LocalTransactionTime	6	int	M	E	M	E	M	E	M	E	M	E	M	E
PrimaryTrack	Var	string	M		O		O		M					
PrimaryPIN	Var	string	O		O		O		O					
SecondaryTrack	Var	string	O		O		O		O					
SecondaryPIN	Var	string	O		O		O		O					
CustomerData	Var	Dictionary[string, string]	O	C	O		O		O					
TransactionExtendedData	Var	string												
OoriginalData ---> TransactionCode ---> LocalTransactionDate ---> LocalTransactionTime	Var	Dictionary[string, string]												
											M1		M1	
											C3		C3	
											C3		C3	
AuthorizationCode	Var	string		M	M2	E	O	E	M	E	C3	E	C3	E
InvoiceNumber	Var	string		E										
ResponseCode	5	string		M		M		M	M	M		M		M
ResponseText	20	string		M		M		M		M		M		M
ReceiptData	Var	string												

Field Name	Size	Type	Sale		Offline		Refund		Cancellation		Void	
			Request	Response	Request	Response	Request	Response	Request	Response	Request	Response
ApplicationType	3	string	M	E	M	E	M	E	M	E1	M	E1
ProcessingMode	1	string	M	E	M	E	M	E	M	E1	M	E1
MessageFormatVersion	3	string	M	E	M	E	M	E	M	E1	M	E1
TerminalIdentification	Var	string	M	E	M	E	M	E	M	E1	M	E1
DeviceTypeIdentifier	1	string	M	E	M	E	M	E	M	E1	M	E1
SystemModel	10	string	M		M		M		M		M	
SystemVersion	10	string	M		M		M		M		M	
TransactionCode	3	string	200	210	205	210	203	210	400	410	220	230
AccountType	1	string	M	E	M	E	M	E	M	E1	M	E1
EntryMethod	1	string	M	E	M	E	M	E	M	E1	M	E1
ServiceCode	1	string										
PumpNumber	2	string		E		E		E		E1		E1
ProductCode	4	string	C2		C2		C2					
ProductUnitPrice	Var	decimal	O		O		O					
ProductNetAmount	Var	decimal										
ProductTaxes	Var	Dictionary[string, decimal]	O		O		O					
ProductAmount	Var	decimal	C1		C1		C1					
ProductQuantity	Var	decimal	C1		C1		C1					
TransactionNetAmount	Var	decimal										
ProductData	Var	List[AtionetProduct]	O		O		O					
TransactionAmount	Var	decimal	C1		C1		C1					
UnitCode	Var	string	O		O		O					
CurrencyCode	3	string										
BatchNumber	Var	int										
ShiftNumber	Var	string										
TransactionSequenceNumber	Var	int	M	E	M	E	M	E	M1	E	M1	E
LocalTransactionDate	8	int	M	E	M	E	M	E	M	E	M	E
LocalTransactionTime	6	int	M	E	M	E	M	E	M	E	M	E
PrimaryTrack	Var	string	M		M		M					

PrimaryPIN	Var	string	O		O		O					
SecondaryTrack	Var	string	O		O		O					
SecondaryPIN	Var	string	O		O		O					
CustomerData	Var	Dictionary[string, string]	O		O		O					
TransactionExtendedData	Var	string										
OriginalData ---> TransactionCode ---> LocalTransactionDate ---> LocalTransactionTime	Var	Dictionary[string, string]										
								M1		M1		
								C3		C3		
								C3		C3		
AuthorizationCode	Var	string		M		M	M	E	C3	E	C3	E
InvoiceNumber	Var	string										
ResponseCode	5	string		M	M	M		M		M		M
ResponseText	20	string		M		M		M		M		M
ReceiptData	Var	string										