The intelligent choice.

iQ Retail
INTEGRATED ACCOUNTING SOFTWARE
A KERRIDGE COMMERCIAL SYSTEMS COMPANY

POS
ACCOUNTING
PAYROLL
HOSPITALITY
CRM
ERP

IQ Retail API & REST Server

Technical User Guide / Documentation
Version 2.0.0 – Dated 11 April 2019

Prepared By:

Danie van Eeden / Dewald Reinke / Gerrit Knoetze

25 Quantum Road | Moonstone Building | First Floor | Technopark | Stellenbosch | 7600
Tel: +27 21 880 0420 | Fax: +27 21 880 0488 | Email: info@iqretail.co.za | www.iqretail.co.za
VAT Number: 4760205510 | Company Reg. 2000/020305/07

Contents

Introduction

This document discusses the IQ API interface and serves as a technical specification document for developers who wish to make use of the IQ API DLL Library and/or REST Server. The IQ API is made available as a 32-bit Native Dynamic Link Library (Microsoft Windows Platform) and exposes methods to client applications that include this library. The API Server is a standalone Windows Service that exposes the same methods via a REST Server instance.

DLL File Name: IQEntAPI.DLL

Bit: 32 Bit

Product Version Relevance: IQ Retail Family of Products from version 6.0.2.0 and higher.

REST Server File Name: IQEntAPIRestServer.exe

Bit: 32 Bit

Product Version Relevance: IQ Retail Family of Product from version 2019.2.0. and higher.

Users of this Guide

This Guide is intended for any third party who wishes to make use of the IQ API exposed methods. This guide assumes full knowledge of the following:

a) Delphi Programming / C# Programming or any other windows based language
b) XML, JSON and XML Schemas
c) Dynamic Link Libraries
d) Memory Management

IQ Retail (Pty) Ltd will make every attempt to update this specification document / guide as the development of the IQ API Library progresses.

General Definitions

IQ API DLL / IQ API: This refers to the IQEntAPI.DLL Dynamic Link Library. This library can be used by third party applications to execute exposed functions provided within the library. At this time, the Library is a Windows Native 32-Bit DLL and can only be used / loaded / accessed via 32-bit processes. You will not be able to load this library from a 64-bit process at this time. The development roadmap of the API contains extended / alternative implementations which may address this problem (ie. COM Objects / Web Services).

IQ REST API: This refers to the IQEntAPIRestServer.exe server. This windows service can be used by third party applications to execute exposed functions provided with the IQ API. At this time, the Executable is a Windows Service 32 Bit application with build in HTTP Server.

IQ Software Application: This refers to the IQ Retail family of products that have been marked with version 6.0.2.0 or higher. The participation of any of these software applications in the API DLL interface is being maintained on continuous bases and will change as per requirements identified. The IQ API Version number has been amended to match the IQ Retail executable version number. Due to close interaction between the DLL and IQ Retail data both these binaries need to be on the same version (determined by the first three sets of the version number eg. **7.0.0**.3). If this is not the case, the IQ API will return a **critical** error due to version incompatibility.

IQ Company: This refers to a parameter / XML Node within an XML Formatted request. This parameter represents the Company ID (number) of a logical / trading company within the IQ Software Application.

Host Application: This refers to the application that loads / makes use of the IQ API.

String: String (in programming terms) would refer to the string type used in the IQ API. This string type is of type AnsiString and consists of 8-bit ANSI characters (Char Type).

Global Type Definitions

Integer: This is signed 32-bit whole number with a range between -2147483648 and 2147483647.

Char: This is an 8-bit data item.

TIQ_Type_Char_Param : Pointer to a Char. This type is used in conjunction with an exposed method and its parameter list to **receive** string type information from the host application.

TIQ_Type_Char_Result : Pointer to a Char. This type is used in conjunction with an exposed method and its parameter list to **return** string type information **to** the host application.

TIQ_Type_Param_Length: This is an Integer type. This type is used in conjunction with TIQ_Type_Char_Param and indicates the length of the parameter value provided by the host application. The IQ API will only consider characters up to the specific length.

TIQ_Type_Param_Length: This is an Integer type. This type is used in conjunction with TIQ_Type_Char_Result and indicates the length of the parameter value **returned to** the host application. The host application should only consider string data up to the specified length.

TIQ_Type_Result: This is an Integer type. This is mainly used to return the result of an exposed method to the Host Application / Calling method.

Calling Conventions: Use the **stdcall** calling convention for all exposed methods.

*Note that all definitions for exposed method / type definitions related to these methods will be given in the relevant section containing details on such exposed method.

Error Type Definitions

The following are definitions of error codes that could be returned by an exposed method call. In the event that a method call fails, the result of such method will contain the error code as a **TIQ_Type_Result** type. This value maps to the following constant declarations in the IQ API Library. In the event of a successful API Call, zero (0) is returned.

Error Code	Error Enumeration	Description	Notes
0	EAPIError_None	No Error	
1	EAPIError_DB_Session	DBISAM Session Error	Occurs when unable to establish active session to DBISAM server.
2	EAPIError_DB_Connect	DBISAM Database Error	Occurs when unable to establish active connection to required database.
3	EAPIError_DB_DBVerify	DBVerify Error	Occurs when unable to successfully read/write from required database.
4	EAPIError_Params_Null	IQEntAPI Parameters Error	Occurs when IQ API receives an empty parameter string from the host application.
5	EAPIError_Table_Open	DBISAM Table Error	Occurs when unable to successfully open a DBISAM table.
6	EAPIError_XML	IQEntAPI XML Error	Occurs when the XML Request data has been incorrectly formatted.
7	EAPIError_Critical	IQEntAPI Critical Internal Error	Used when an unhandled exception / internal critical error occurs.
8	EAPIError_Processing	IQEntAPI Processing Error	Used when an error occurred during the requested processing.
9	EAPIError_Processing_Data	IQEntAPI Processing Error Data	Used to indicate that the processing has completed successfully BUT that there are Error Items as a result of processing / found after processing.
10	EAPIError_Instructions	IQEntAPI Instructions Execution Error	Used to indicate that the API was unable to process and Additional

			Instruction (part of the IQ_Instruction_Set node).

DLL Exposed Methods will return all Error Data as XML or JSON Formatted Text via the return parameter of type TIQ_Type_Char_Result .

XML & JSON Format, visual diagram and brief description follows:

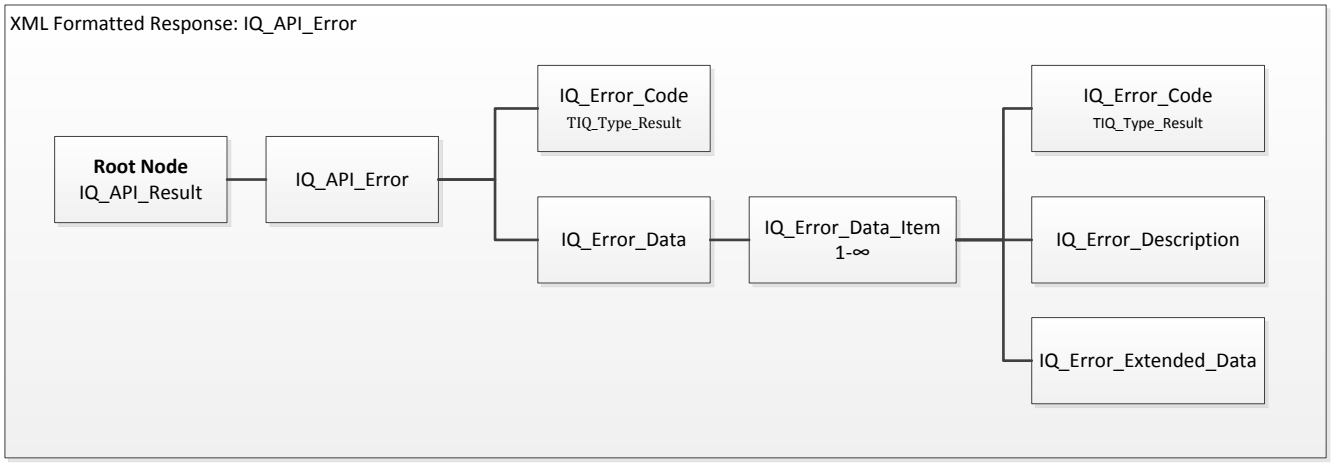
XML Format:

```
<?xml version="1.0" encoding="UTF-8"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>5</IQ_Error_Code>
    <IQ_Error_Description>Record Count Zero. No Records To
                Retrieve</IQ_Error_Description>
    <IQ_Error_Data>
     <IQ_Error_Data_Item>
      <IQ_Error_Code>5</IQ_Error_Code>
      <IQ_Error_Description>Record Count Zero. No Records To Retrieve
      </IQ_Error_Description>
      <IQ_Error_Extended_Data />
    </IQ_Error_Data_Item>
   </IQ_Error_Data>
 </IQ_API_Error>
</IQ_API_Result>
```

JSON Format:

```
{
  "IQ_API_Error":[
    {
      "IQ_Error_Code":5,
      "IQ_Error_Description":"Record Count Zero. No Records To Retrieve",
```

```
    "IQ_Error_Data":{
      "IQ_Error_Data_Items":[
        {
          "IQ_Error_Code":5,
          "IQ_Error_Description":"Record Count Zero. No Records To Retrieve"
        }
      ]
    }
  }
 ]
}
```

XML Visual Diagram

Brief Explanation

In the event of an error being returned by the IQ API, the exposed function will return, as its result of TIQ_Type_Result, the last related error code. If more than one error occurred, this value will be that of the last error. This code can be compared against the above Error Type Definitions.

If the Error Code is 0 (zero) (ie. No Error Occurred), the nodes named IQ_Error_Data will not be populated with any data. If the Error Code returned is anything other than 0 (zero), the IQ_Error_Data will contain the related IQ_Error_Data_Item nodes. Each of these nodes will contain

a) IQ_Error_Code – The Error number as per above Error Type Definitions

b) IQ_Error_Description – An error description relevant to the problem that has occurred. This can be used for dialog messages or logging information in the host application.
c) IQ_Error_Extended_Data – This node can contain any form of XML data that is relevant to the IQ_Error_Code (eg. If the host application fails in its attempt to submit Stock Data Attributes via the IQ API, this node will contain the unsuccessful Stock Item XML Data.

In addition to the relevant Error related data, the API may also return Success related data (in addition to a zero error code).

a) IQ_API_Success – This node contains details regarding successful submission of data. Currently this node supports success information for Document submissions only.

-END OF SECTION – ERROR TYPE DEFINITIONS-

API Environment Pre-requisites

1. The IQ API DLL (IQEntAPI.DLL) and IQ REST API must be located within the IQEnterprise folder (containing the IQ Retail software executable named IQEnterprise.exe).
2. The IQ Retail software must be licensed to make use of the IQ API Library. This can be requested via the IQ Retail software (Menu Item -> Support -> Registration).
3. The IQ Retail software must contain at least one (1) system user that has been set up for API transaction purposes. This can be done via the menu Utilities -> Menu Access and Security -> User and Group Maintenance. From here a system user must be amended to allow usage via the API Library. The API caller will specify the user number of this user account and the related API Password as setup from within this setup module. Any transactions occurring as a result of an API call will be with emulation of this user number.

-END OF SECTION – PRE-REQUISISTS-

Enumerated Type Definitions

TIQ_API_Filter_Operator

This enumerated type forms part of an XML Formatted Request (within an **XML_Filter_Part** Node) and is used to indicate the logical operator for the Filter being applied. The following values are legal values for this node (see highlighted column).

Operator ID	Operator in XML	Description	Logical Equivalent	Comments
0	opNone	Not Applicable	Not Applicable	
1	opEquals	Equals	=	Binary
2	opNotEquals	Not Equal	<>	Binary
3	opLess	Less Than	<	Binary
4	opLessEqual	Less Than or Equal	<=	Binary
5	opMore	More Than	>	Binary
6	opMoreEqual	More Than or Equal	>=	Binary
7	opLike	Like	Like	Binary
8	opNotLike	Not Like	Not Like	Binary
9	opBlank	Contains no value	NULL	Unary
10	opNotBlank	Contains some value	Not NULL	Unary
11	opBetween	Between	Between	N-Ary
12	opNotBetween	Not Between	Not Between	N-Ary
13	opIn	Within	In	N-Ary
14	opNotIn	Not Within	Not In	N-Ary

TIQ_API_Filter_Concat_Operator

This enumerated type forms part of an XML Formatted Request (within an **XML_Filter** Node) and is used to indicate which logical operator should be used to concatenate two or more **XML_Filter_Part** Nodes.

Operator ID	Operator in XML	Logical Equivalent

15	opOR	OR
16	opAND	AND

TIQ_API_Request_DB_Filter_Type

This enumerated type forms part of an XML Formatted Request (within an **XML_Filter** Node) and is used to indicate which type of Filter Specification method will be used within this XML Request.

Filter Type ID	**Value in XML**	**Description**
0	EAPIFilter_None	Not Applicable
1	EAPIFilter_XML	XML Based Filter Specification
2	EAPIFilter_TEXT	TEXT Based Filter Specification

TIQ_API_Filter_Operand_Type

This enumerated type forms part of an XML Formatted Request (within an **XML_Filter_Part** Node) and is used to indicate that Data Type of the provided Operand Value. This is used to distinguish between text, numbers, logicals and enumerated values and determines how the value is interpreted by the IQ API.

ID	XML Value	Interpretation of Operand Value	Comments
0	opTypeNone	Not Applicable	
1	opTypeString	String / Text Value	
2	opTypeInteger	Integer / Whole Number	
3	opTypeFloat	Floating Point Number	
4	opTypeBoolean	TRUE or FALSE	
5	opTypeEnum	Based on TIQ_API_Filter_Operand_Enum_Type	TIQ_API_Filter_Operand_Enum_Type is converted to corresponding Integer value before comparison.
6	opTypeDataField	Another Data Attribute	Attribute is compared to another data field.

TIQ_API_Filter_Operand_Enum_Type

This enumerated type forms part of an XML Formatted Request (within an **XML_Filter_Part** Node) and is used to indicate which Enumerated String has been specified within the Operand_Value node. The value within the Operand_Value node will be converted to the corresponding Native IQ Value which, in turn, will be used for the filter comparison. (ie. All categories below refer to some attribute of a Data table in the IQ Software Systems that has a native storage value that can be mapped to a “Friendly” display value. The calling application specifies the “friendly” value within the XML Request, together with an indication of which category it represents, and the IQ API will then convert the “friendly” value to a native IQ value before applying the filter comparison). Friendly values are specified in the relevant XML Schemas.

ID	XML Value	Description	Comments
0	opEnum_None	Not Applicable	
1	opEnum_Stock_Category	Stock Category	

2	opEnum_Stock_Volumetrics	Volumetrics	
3	opEnum_Stock_AutoCalc	Stock Price Calculation Method	
4	opEnum_Stock_MarkupIndex	Stock Markup Index	
5	opEnum_Stock_OrderMethod	Stock Ordering Method	
6	opEnum_Stock_ABCClass	Stock ABC Classification	
7	opEnum_Stock_VatRate	Stock Vat Rate	

TIQMasterError

These enumerated types are used in conjunction for Master File Imports / Master Data submissions. These enumerated types can be used to identify import errors during a Master file submission and can be used to specify Override Items that should be considered during the import process. These error types apply to the following Master imports: Stock, Debtors, Creditors, Major Departments, Minor Departments, Ranges, Categories, Promotions, Store Departments, Stock Contract Pricing, Stock PriceLists.

ID	XML Value	Description	Relevance	Can Override
0	imeNone	Not Applicable	None	No
1	imeCritical	Critical Internal Error	All	No
2	imeZeroRecords	Zero Records Available for Import	All	No
3	imeInvalidMasterType	Invalid Master Type Indicator	All	No
4	imeInvalidEnumString	Invalid Enumerated String	All	No
5	imeInvalidCode	Invalid Stock Code	Stock Master	No
6	imeInvalidDepartment	Invalid Major Department	Stock Master	No
7	imeInvalidColour	Invalid Stock Colour Number	Stock Master	No
8	imeInvalidLineColour	Invalid Stock Line Colour Number	Stock Master	No
9	imeInvalidRegSupplier	Invalid Main / Regular Supplier	Stock Master	No

10	imeVatRate	Invalid Vat Rate	Stock Master	No
11	imeInvalidSellPrice	Invalid Selling Price / Format	Stock Master	No
12	imeDuplicateBarcode	Duplicate Barcode already exists	Stock Master	No
13	imeDuplicateCode	Duplicate Stock Code already exists	Stock Master	Yes
14	imeInvalidOrderMethod	Invalid Ordering Method	Stock Master	No
15	imeInvalidOrderFormula	Invalid Ordering Formula	Stock Master	No
16	imeInvalidAVGCost	Invalid Average Cost / Format	Stock Master	No
17	imeInvalidMaxDiscount	Invalid Maximum Discount Value / Format	Stock Master	No
18	imeInvalidItemCircularRef	Problem with regards to Cascading Items	Stock Master	No
19	imeInvalidReportCFactor	Invalid Conversion Factor for cascading items	Stock Master	No
20	imeInvalidCascadingCost	Cascading item has cost	Stock Master	No
21	imeInvalidSize	Invalid Stock Size	Stock Master	No
22	imeInvalidAccount	Invalid Debtor / Creditor Account	Dr / Cr Master	No
23	imeDuplicateAccount	Duplicate Debtor / Creditor Account	Dr / Cr Master	Yes
24	imeInvalidGroupAccount	Invalid Group / Link Account	Dr / Cr Master	No
25	imeInvalidCurrency	Invalid Currency Identifier	Dr / Cr Master	No
26	imeInvalidSettlementDiscount	Invalid Settlement Discount Value / Format	Dr / Cr Master	No
27	imeInvalidLayout	Invalid Document Layout for Account	Dr / Cr Master	No
28	imeInvalidRoute	Invalid Delivery Router for Account	Dr / Cr Master	No
29	imeInvalidRep	Invalid Sales Rep for Account	Debtor Master	No
30	imeInvalidRiskProfile	Invalid Risk profile for interest calculation	Debtor Master	No
31	imeInvalidIncDiscPerc	Invalid Invoice Discount Percentage Value / Format	Dr / Cr Master	No
32	imeReportItemOnhand	Cascading item has onhand value	Stock Master	No
33	imeInvalidCascadingCode	Problem with cascading items	Stock Master	No
34	imeCascadingDisable	Cascading items not enabled	Stock Master	No

35	imeInvalidNotification	Invalid Account Notification	Dr / Cr Master	No
36	imeInvalidPriceList	Invalid Pricelist for account	Dr / Cr Master	No
37	imeInvalidAccountGroup	Invalid Account Group	Dr / Cr Master	No
38	imeInvalidStyle	Invalid Stock Style	Stock Master	No
39	imeInvalidCategory	Invalid Stock Category	Stock Master	No
40	imeInvalidRange	Invalid Stock Range	Stock Master	No
41	imeInvalidDeptCode	Invalid Department Code	Major Dept.	No
42	imeDuplicateDept	Existing Department	Major Dept.	Yes
43	imeInvalidLedgerAcc	Invalid Ledger Account	Major Dept.	No
44	imeInvalidLedgerDepartment	Invalid Ledger Department	Major Dept.	No
45	imeInvalidPostingMethod	Invalid Posting Method	Major Dept.	No
46	imeInvalidLineRep	Invalid Line Rep	Major Dept.	No
47	imeInvalidGroupCode	Invalid Sub Department Code	Minor Dept.	No
48	imeDuplicateGroup	Existing Sub Department	Minor Dept.	Yes
49	imeInvalidCategoryCode	Invalid Category Code	Categories	No
50	imeDuplicateCategory	Existing Category	Categories	Yes
51	imeInvalidRangeCode	Invalid Range Code	Ranges	No
52	imeDuplicateRange	Existing Range	Ranges	Yes
53	imeInvalidStoreDepartment	Invalid Store Department	Dr / Cr Master	No
54	imeInvalidPromo	Invalid Promotion Code	Promotions	No
55	imeDuplicatePromo	Existing Promotion	Promotions	Yes
56	imeInvalidEndDate	Invalid End Date	Promotions	No
57	imeInvalidQuantity	Invalid Quantity	Promotions	No
58	imeInvalidItem	Invalid Stock Item	Promotions	No
59	imeInvalidPrice	Invalid Item Price	Promotions	No
60	imeDuplicateItem	Duplicate Stock Item	Promotions	No
61	imeDuplicateDepartment	Existing Store Department	Store Departm.	Yes
62	imeInvalidCreatedDate	Invalid Created Date	Store Departm.	No

63	imeDuplicateContractPrice	Existing Contract Price	Contract Price	Yes
64	imeInvalidContractPrice	Invalid Contract Price	Contract Price	No
65	imeInvalidDebtorGroup	Invalid Debtor Group	Contract Price	No
66	imeInvalidProductDept	Invalid Product Department	Contract Price	No
67	imeInvalidDiscount	Invalid Discount Percentage	Contract Price	No
68	imeInvalidContractType	Invalid Contract Type	Contract Price	No
69	imeDuplicateRep	Existing Sales Representative	Sales Reps	Yes
70	imeInvalidCommPerc	Invalid Commission Percentage	Sales Reps	No
71	imeInvalidCommType	Invalid Commission Type	Sales Reps	No
72	imeInvalidEmailStatus	Invalid Email Status	Sales Reps	No
73	imeRepDoesNotExist	Rep Code Does not Exist	Sales Reps	No
74	imeDuplicatePriceList	Existing Stock Price List Number	Price Lists	Yes
75	imeInvalidPriceType	Invalid Price Type	Price Lists	No

TIQDocumentError

These enumerated types are used in conjunction for Document Imports / data submissions. These enumerated types can be used to identify import errors during a Document submission and can be used to specify Override items that should be considered during the import process. These error types apply to the following Document Types: Sales Orders, Purchase Orders, Job Cards, Quotes, Invoices.

ID	XML Value	Description	Relevance	Can Override
0	ideNone	Not Applicable	None	No
1	ideCritical	Critical Internal Error	All	No
2	ideZeroRecords	Zero Records Available for Import	All	No
3	ideInvalidDocumentType	Invalid Document Type	All	No
4	ideInvalidAccount	Invalid Account	All	No
5	ideInvalidDate	Invalid Document Date / Format	All	No
6	ideInvalidRep	Invalid Sales Rep	SOR, Quote, Job	Yes

7	ideInvalidCashier	Invalid Cashier	All	Yes
8	ideInvalidStaff	Invalid Staffcode	All	No
9	ideStockOnHold	Stock Item On Hold	All	Yes
10	ideAccountOnHold	Account On Hold	All	Yes
11	ideDataError	Data Read Error	All	No
12	ideDuplicateDocument	Duplicate Document	All	No
13	ideStockInvalid	Invalid Stock Code	All	No
14	ideInvalidCurrency	Invalid Currency Indicator	All	No
15	ideTotalIncorrect	Document Total Incorrect	All	Warning Only
16	ideInvalidDateRange	Invalid Date Range	All	Yes
17	ideSerialsNotEnabled	Serial Number feature not enabled	All	Yes
18	ideStrictSerialsEnabled	Strict Serial Number Control Enabled	All	No
19	ideNegativeStock	Negative Stock Levels	All	Yes
20	ideInvalidDiscount	Invalid Discount Amount/ Format	All	No
21	ideVolumetricsEnabled	Volumetrics Enabled	All	No
22	ideCascadingLinksInvalid	Problem with cascading items	All	No
23	ideExtraChargeNotEnabled	Extra Charges Not Enabled	All	Warning Only
24	ideInvalidLineRep	Invalid Line Sales Rep	SOR, Quote, Job	No
25	ideRepOnHold	Sales Rep on Hold	SOR, Quote, Job	Yes
26	ideInvalidDepartment	Invalid Stock Department	All	No
27	ideInvalidLoyalty	Invalid Loyalty Account	INV, CRN	No
28	ideInvalidLoyaltyType	Invalid Loyalty Type – Offline Only	INV, CRN	No
29	ideInvalidTenderMedia	Tender Media Invalid	INV, CRN	No
30	ideInvalidTenderAmount	Tender Amount Invalid	INV, CRN	No
31	ideStockBatches	Stock Batch Control	INV, CRN	No
32	ideInvalidStoreDepartment	Invalid Store Department	All	No

33	ideBlankOrderNumber	Blank Order Numbers for Debtors	INV	Yes

TIQJournalError

These enumerated types are used in conjunction for Debtor and Creditor Journal Imports / data submissions. These enumerated types can be used to identify import errors during a Journal submission and can be used to specify Override items that should be considered during the import process. These error types apply to Debtor and Creditor Journal imports.

ID	XML Value	Description	Relevance	Can Override
0	ijeNone	Not Applicable	None	No
1	ijeCritical	Critical Internal Error	All	No
2	ijeZeroRecords	Zero Records Available for Import	All	No
3	ijeInvalidJournalType	Invalid Journal Type	All	No
4	ijeInvalidAccount	Invalid Account	All	No
5	ijeInvalidDate	Invalid Journal Date / Format	All	No
6	ijeInvalidBranch	Invalid Branch	All	No
7	ijeInvalidDept	Invalid Department	All	No
8	ijeInvalidRep	Invalid Sales Representative	Debtor Journals	No
9	ijeInvalidCode	Invalid Transaction Code	All	No
10	ijeInvalidSplitLdgrAccount	Invalid Split Ledger Account	All	No
11	ijeInvalidSplitBranch	Invalid Split Branch	All	No
12	ijeInvalidSplitDept	Invalid Split Department	All	No
13	ijeInvalidSplitBalance	Invalid Split Balance	All	No
14	ijeInvalidSplitVatRate	Invalid Split Vat Rate	All	No

TIQSystemError

These enumerated types are used in conjunction for System Security Imports / data submissions. These enumerated types can be used to identify import errors during a System / Security submission and can be used to specify Override items that should be considered during the import process. These error types apply to System Security Imports (Groups and Users).

ID	XML Value	Description	Relevance	Can Override
0	iseNone	Not Applicable	None	No
1	iseCritical	Critical Internal Error	All	No
2	iseInvalidSecurity	Invalid Security Code	All	No
3	iseInvalidSystemType	Invalid System Type	All	No
4	iseZeroRecords	Zero Records Available	All	No
5	iseInvalidGroup	Invalid Group Code	All	No
6	iseDuplicateGroup	Existing group	All	Yes
7	iseInvalidUser	Invalid User Code	All	No
8	iseDuplicateUser	Existing User	All	Yes
9	iseInvalidPin	Invalid PIN	All	No
10	iseInvalidForcePass	Invalid Password Force	All	No
11	iseInvalidAskForPin	Ask For Pin – State Invalid	All	No
12	iseInvalidCompany	Invalid Company ID	All	No

TIQEntAPI_Related_Data_Type

These enumerated types are used in conjunction with the **Related Data** feature for XML request calls.

ID	XML Value	Description
0	api_rd_None	Not Applicable
1	api_rd_Debtors_Reps	Debtor Sales Reps
2	api_rd_Debtors_ASM	Debtor Area Sales Managers
3	api_rd_UserData	Debtor User Defined Data

TIQEnt_API_Instruction_Type

These enumerated types are used in conjunction with the **IQ Instruction** feature for XML submission calls.

ID	XML Value	Description
0	apiitNone	Not Applicable
1	apiitPrint	Print Document
2	apiitEmailPDF	Email Document
3	apiitExportPDF	Export Document
4	apiitCustomReport	Custom Report
5	apiitSubmitConfirmation	Submission Confirmation
6	apiitSubmitError	Submission Error

TIQEnt_API_Document_Type

The following enumerated types represent both documents and transaction types within the IQ Retail system and are used in conjunction with numerous features / facilities. The column API Supported indicates which if these are currently supported via API calls.

ID	XML Value	Description	API Supported
0	iqNone	Not Applicable	Yes
1	iqInv	Invoice	Yes
2	iqInvSlip	Invoice (Slip)	No
3	iqCRN	Credit Note	Yes
4	iqGRV	Goods Receiving	Yes

5	iqRTS	Returned Goods	Yes
6	iqQuote	Quote	Yes
7	iqQuoteEdit	Quote	No
8	iqPOS	Point of Sale	No
9	iqSOR	Sales Order	Yes
10	iqPORInvoice	Purchase Order	No
11	iqSORInvoice	Sales Order	No
12	iqDelNote	Delivery Note	No
13	iqJob	Job card	Yes
14	iqJobInvoice	Job Card	No
15	iqTransfer	Stock Transfer	No
16	iqKKTransfer	Stock Transfer	No
17	iqAgent	Agent Transaction	No
18	iqBill	Bill of Manufacturing	No
19	iqConsign	Consignment	No
20	iqRecallQuote	Quote	No
21	iqLoyalty	Loyalty	No
22	iqPickSlip	Picking Slip	No
23	iqINVQuote	Invoice	No
24	iqStatement	Statement	No
25	iqAdviceNote	Advice Note	No
26	iqReceipt	Debtors Receipt	No
27	iqCRPaument	Creditors Payment	No
28	iqBillDisassemble	Bill of Manufacturing	No
29	iqPOS	Point of Sale	No
30	iqTFIn	Stock Transfer In	No
31	iqTFOut	Stock Transfer Out	No
32	iqBillIn	Bill of Manufacturing IN	No
33	iqBillOut	Bill of Manufacturing OUT	No
34	iqPettyOut	Petty Cash Out	No

35	iqPettyIn	Petty Cash In	No
36	iqDrJournal	Debtors Journal	No
37	iqCRJournal	Creditors Journal	No
38	iqPOSCashSale	Point of Sale – Cash Sale	No
39	iqPOSCashRefund	Point of Sale – Cash Refund	No
40	iqPOSLaybyeNew	Point of Sale – New Laybye	No
41	iqPOSLaybyePayment	Point of Sale – Laybye Payment	No
42	iqPOSLaybyeRefund	Point of Sale – Refund	No
43	iqReceiptRefund	Debtors Receipt – Refund	No
44	iqTFRequest	Transfer Request	No
45	iqPOSReceipt	Point of Sale – Receipt	No
46	iqSundryIR	Sundry Issues / Receipts	No
47	iqSundryIssue	Sundry Issue	No
48	iqSundryReceipt	Sundry Receipt	No
49	iqFPO	Orders	No
50	iqPOSCashTFIn	Point of Sale – Cash Transfer IN	No
51	iqPOSCashTFOut	Point of Sale – Cash Transfer OUT	No
52	iqPOSAccountSale	Point of Sale – Account Sale	No
53	iqPOSAccountRefund	Point of Sale - Account Refund	No
54	iqRFQ	Request For Quote	No
55	iqSDN	Supplier Delivery Note	No
56	iqSystemGroup	System Group	No
57	iqSystemUser	System User	No
58	isClaim	Claim	No
59	iqDRStmt	Debtor Statement	No

TIQEnt_API_Email_Recipient_Type

The following enumerated types are used in conjunctions with **IQ Instructions** and indicate the type of recipient for email addresses specified.

ID	XML Value	Description
0	api_rt_None	Not Applicable
1	api_rt_To	To (Normal Recipient)
2	api_rt_CC	CC (Carbon Copy)
3	api_rt_BCC	BCC (Blind Carbon Copy)

-END OF SECTION – ENUMERATED TYPE DEFINITIONS-

API Important Notes

*** Memory Allocation ***

Due to the nature of memory allocation, and the differences between various compilers in this regard, the following important rules and comments need to be understood and adhered by throughout the use of the IQ API.

Memory Allocation for Result Buffer:

All exposed IQ API methods that return a PChar result (ie. a series of characters / string type information) will allocate the required memory for the series of characters. This memory allocation needs to happen within the IQ API DLL seeing as though the host application will not be able to determine the buffer size to allocate beforehand. The memory allocation by the IQ API DLL will happen automatically and should require no action by the host application. The IQ API DLL will also return the related buffer length (length of the PChar Result). The host application can extract the relevant information by starting at the first position in the result and extracting data up the provided length.

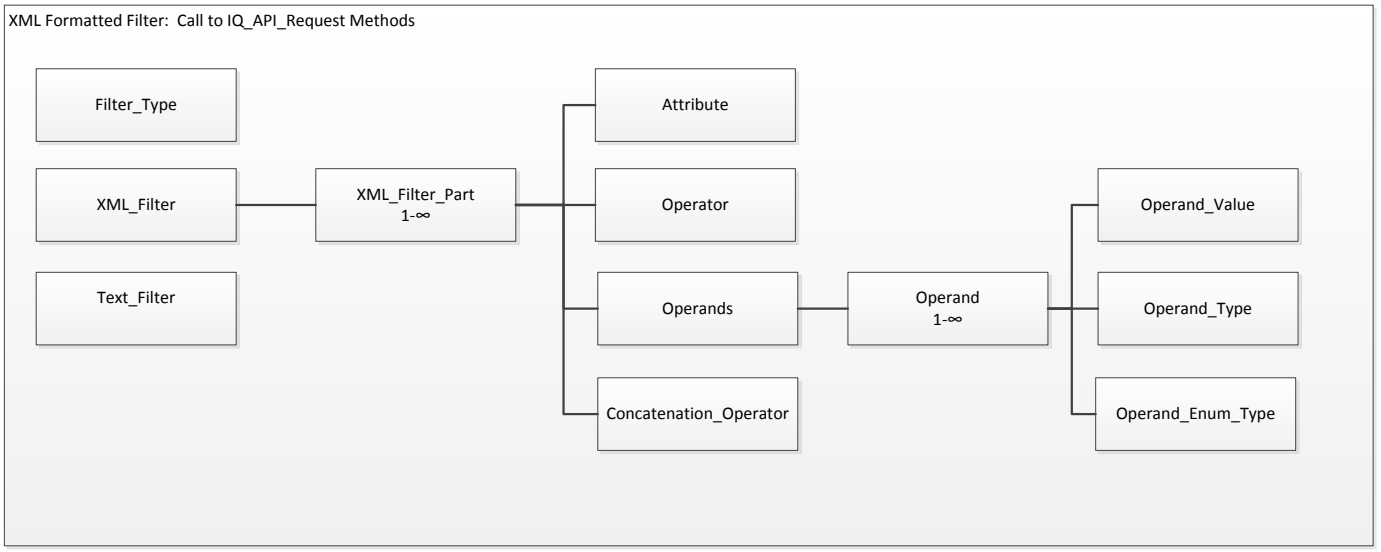
Releasing Memory of Result Buffer:

In an attempt to minimize any memory related errors the IQ API DLL must be the process to free the allocated memory for the result buffer. The host application should NOT attempt to release the memory of the result buffer by means of native calls. The IQ API DLL provides another exposed method named **IQ_API_Free_PChar**. In order to allow the IQ API DLL to free allocated memory for the result buffer, the host application simply needs to call this function and pass to it, as its only parameter, the PChar. The IQ API DLL will determine if the pointer is assigned and, if so, will release the allocated memory. Failing to call this method will result in memory leaks within the IQ API DLL.

-END OF SECTION – IMPORTANT NOTES-

API Call Features

XML Formatted Filters



XML Formatted Filters form part of XML Formatted Requests. The XML Request allows the user to specify a Record Filter that will be applied to the relevant data before extraction (thus limiting the number of records).The IQ API supports two types of Filters in XML Formatted requests:

a) an XML Formatted Filter – using XML Nodes & "*Friendly*" Names to specify Filter operators and operands.
b) and a Text Filter – using DBISAM SQL formatted text

Node Explanations:

Filter_Type This node contains an indicator of the filter format to be expected in the XML Request. The Filter format can either be a Text Filter OR a Filter consisting of XML Nodes in the required format. The corresponding XML Value for this enumerated type can be found in **Enumerated Type Definitions** for TIQ_API_Request_DB_Filter_Type.

XML_Filter: This node is taken into consideration if the Filter_Type node contains the value **EAPIFilter_XML .**
The XML_Filter node consists of numerous additional child nodes that can be used to build up one or more logical expressions for record filtering purposes. The following subnodes need to exists and contain values as specified.

*XML_Filter_Part: This node is a group node containing additional nodes that specify the construction of the filter expression. Each **XML_Filter_Part** can be concatenated with another **XML_Filter_Part**. The concatenation operator is specified in the relevant node **Concatenation_Operator**.*

*Attribute: This node is a child node of an **XML_Filter_Part** node. It contains the attribute that needs to be evaluated against some condition. The attribute should be specified using the Friendly name (as per XML Schema – see Appendix). Native names are not supported.*

*Operator: This is a child node of an **XML_Filter_Part**. This node contains a logical operator to be used in the evaluation / logical comparison of the Friendly attribute name (specified in the Attribute Node). Logical operators are specified as XML Values which will be mapped automatically to native operators prior to execution. Logical Operators can be found under Enumerated Type Definitions for **TIQ_API_Filter_Operator**.*

*Operands: This is a child node of an **XML_Filter_Part** and is a group node containing multiple Operand nodes. The group can contain between one (1) and many (∞) Operand nodes.*

*Operand: This node is a child of the **Operands** group node. Each Operand node contains additional nodes specifying the Operand details.*

*Operand_Value: This node is a child of an **Operand** node. This node contains the value that the Attribute will be compared against (by using the Operator specified). These operand values can be of different types and such type need to be specified under the Operand_Type node.*

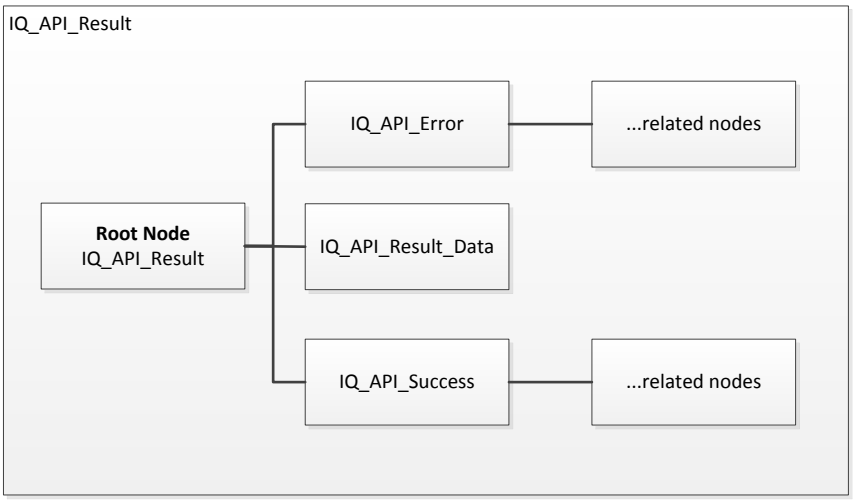
*Operand_Type: This node is a child of an **Operand** node. This node contains the XML Value for **TIQ_API_Filter_Operand_Type** (mappings can be found under **Enumerated Type Definitions**). This XML Value will automatically be converted to the native operand type. This operand type will determine how the Operand_Value is interpreted (ie. As Text, Number, Boolean etc).*

*Operand_Enum_Type; This node is a child node of an Operand Node. This node is only required if the Operand_Type node contains the **opTypeEnum** value. The opTypeEnum value specifies that the Operand_Value refers to a **TIQ_API_Filter_Operand_Enum_Type.** If this is the case, the Operand_Value will be converted according to the Operand_Enum_Type that has been specified. The Operand_Value will contain the Friendly value as per Stock Master XML specification. (eg.if Operand_Enum_Type contains **opEnum_Stock_Categor,** the value "Stock Item" will be converted to the Native Integer value of 1. The value "Non Stock Item" will be converted to the Integer value of 2). As a result the calling application does not need to know the Native values.*

Text_Filter: This node is taken into consideration if the Filter_Type node contains the value **EAPIFilter_TEXT.** The value of this node is a Logical Expression in the supported DBISAM SQL Syntax. Native Field names must be used – Friendly fieldnames are **not** supported.

XML Formatted Response

IQ API calls that return a TIQ_Type_Char_Param type result will return this result as an XML Formatted response. All XML Formatted responses will contain the following layout outline.



IQ_API_Result: This is the root node of the result. It contains the IQ_API_Error , IQ_API_Result_Data and IQ_API_Success subnodes.

IQ_API_Error: This node contains any error related data. See the **Error Type Definitions** for detailed information.

IQ_API_Data_Result: This node contains actual result data. In the event of a successful request to the IQ API, this node will contain the requested data.

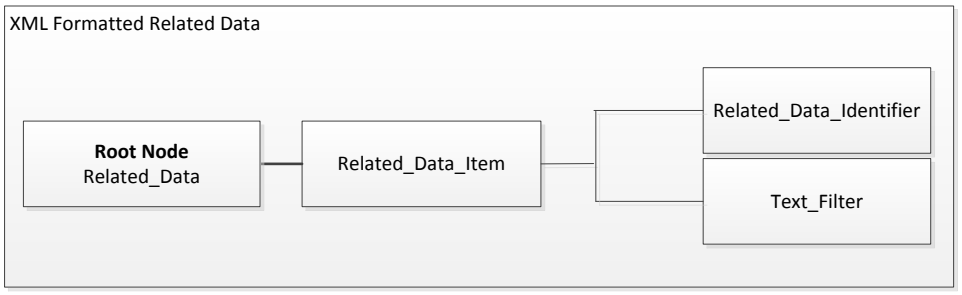
IQ_API_Success: This node contains other, relevant information in the case of a successful call. This is currently only supported for document type submissions and, if relevant, contains the details on successfully submitted documents. See **IQ_API_Success** section for details.

-END OF SECTION – XML FORMATTED RESPONSE-

XML Requests - Related Data

A selected set of API Request calls support an additional feature named **Related Data** (identified by the **Related_Data** node). This call feature allows the caller to request additional related data records from a relevant data table. Internally the API will perform normal SQL Joins to retrieve the data and will present it to the user in XML Format.

The Related_Data node must be created as a child node of the main request node (eg. IQ_API_Request_Debtor) and should consist of the following child nodes:



Related_Data_Item: This node represents a single related data request. The item identifies a set of related data being requested. The **Related_Data** node can contain zero (0) to many (∞) **Related_Data_Item** nodes (each representing a requested set of related data).

Each Related_Data_Item node should consist of the following child nodes:

Related_Data_Identifier: This node represents an enumerated type value that identifies the type of related data being requested. See **TIQEntAPI_Related_Data_Type** under **Enumerated Type Definitions** for value descriptions. This identifier will determine which data tables will be joined to the set of requested data determined by the main XML Node in the API Request.

Text Filter: The field represents a Text Based DBISAM Filter that will be applied to the related set of data. This filter will be concatendated to the filter for the main data request (whether XML or Text). Related Data requests only support TEXT based filters (and as a result Native field names only). Friendly fieldnames are **not** supported.

Notice: Related Data requests will have a performance penalty due to back end joins on the data.

XML Submissions – Instructions

IQ XML Submissions support an additional feature named **IQ Instructions** (identified by the **IQ_Instruction_Set** node). This call feature allows the caller to specify additional instructions to be performed (post submission) by the API. The number of supported instructions are finite and specified under the section **TIQEnt_API_Instruction_Type** under **Enumerated Type Definitions**.

The IQ_Instruction_Set node must be created as a child node of the main submission node (eg. IQ_API_Submit_Document_Sales_Order) ans should consist of the following child nodes (based on the instruction type):

IQ_API_Instruction_Print

Instruction Type: **apiitPrint**

This instruction applies to:

a) Document related submissions (eg. Sales Order, Invoice etc) if the related parameters are excluded.
b) All XML Calls if the related parameters are provided.

The instruction allows the caller to request the API to

a) Submit the document(s) for printing purposes after successful submission of the document(s) to the IQ Database. If the document(s) does not pass validation (and is thus not submitted), the failed document(s) will, as a result, also not be submitted for printing.
b) Submit a request to the API to print a specific document (irrespective of the submitted documents).

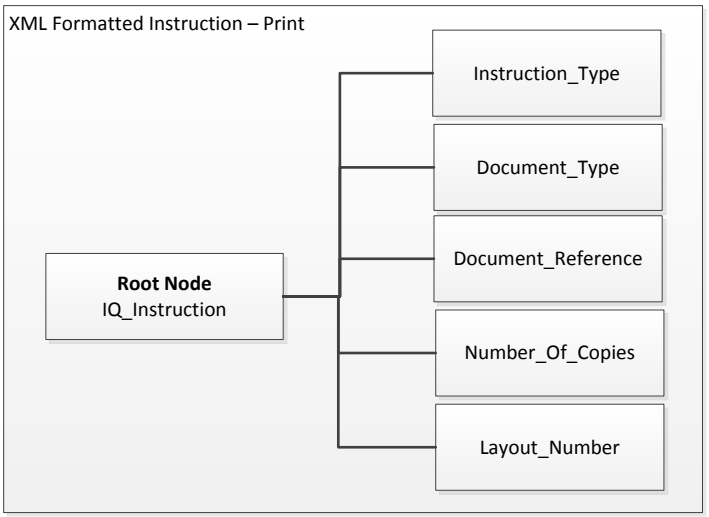
The Print instruction allows for the specification of the following parameters (in the form of child XML nodes).

a) Document_Type: This is represented by an enumerated type (see section **TIQEnt_API_Document_Type** under **Enumerated Type Definitions**) and specifies the type of document to be printed. If this value is not specified, the API will attempt to print all successfully submitted documents that formed part of the API call.

   Supported documents types are
   a. Purchase Orders
   b. Sales Orders
   c. Job Cards
   d. Quotes
   e. Invoices

f. Credit Notes
g. GRVs
h. RTSs

b) Document_Reference: This is only relevant if the Document_Type has also been provided. This node represents a string value that identifies the document to be printed by its Document number.
c) Number_Of_Copies: This is always relevant if provided. The value is of Integer type determines the number of copies to be printed.
d) Layout_Number: This is always relevant if provided. The value is of Integer type and determines the layout to be printed.



IQ_API_Instruction_Export:

Instruction Type: **apiitExportPDF**

This instruction applies to:

a) Document related submissions (eg. Sales Order, Invoice etc) if the related parameters are excluded.
b) All XML Calls if the related parameters are provided.

The instruction allows the caller to request the API to

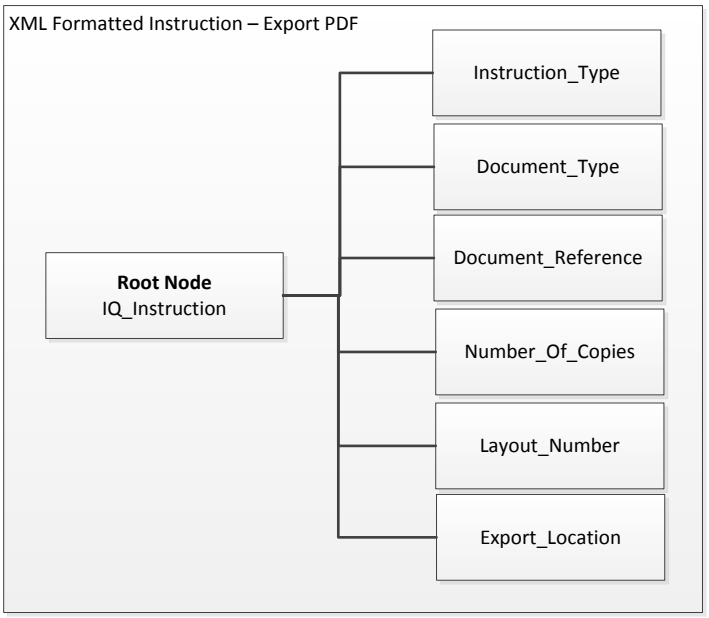
a) Submit the document(s) for exporting purposes after successful submission of the document to the IQ Database. If the document does not pass validation (and is thus not submitted), the failed document(s) will, as a result, also not be submitted for exporting.
b) Submit a request to the API to export a specific document (irrespective of the submitted documents).

Supported documents types are

a. Purchase Orders
b. Sales Orders
c. Job Cards
d. Quotes
e. Invoices
f. Credit Notes
g. GRVs
h. RTSs

The Export instruction allows for the specification of the following parameters (in the form of child XML nodes) in addition to the XML parameters supported by the Print Instruction defined above.

a) Export_Location: This field identifies the location on the IQ Server (folder / directory) where the export files must be placed. This XML Node is optional and, if not provided, the default Export Location (as set up within the IQ Retail software under Company Details) will be used.



IQ_API_Instruction_Email:

Instruction Type: **apiitEmailPDF**

This instruction applies to:

a) Document related submissions (eg. Sales Order, Invoice etc) if the related parameters are excluded.
b) All XML Calls if the related parameters are provided.

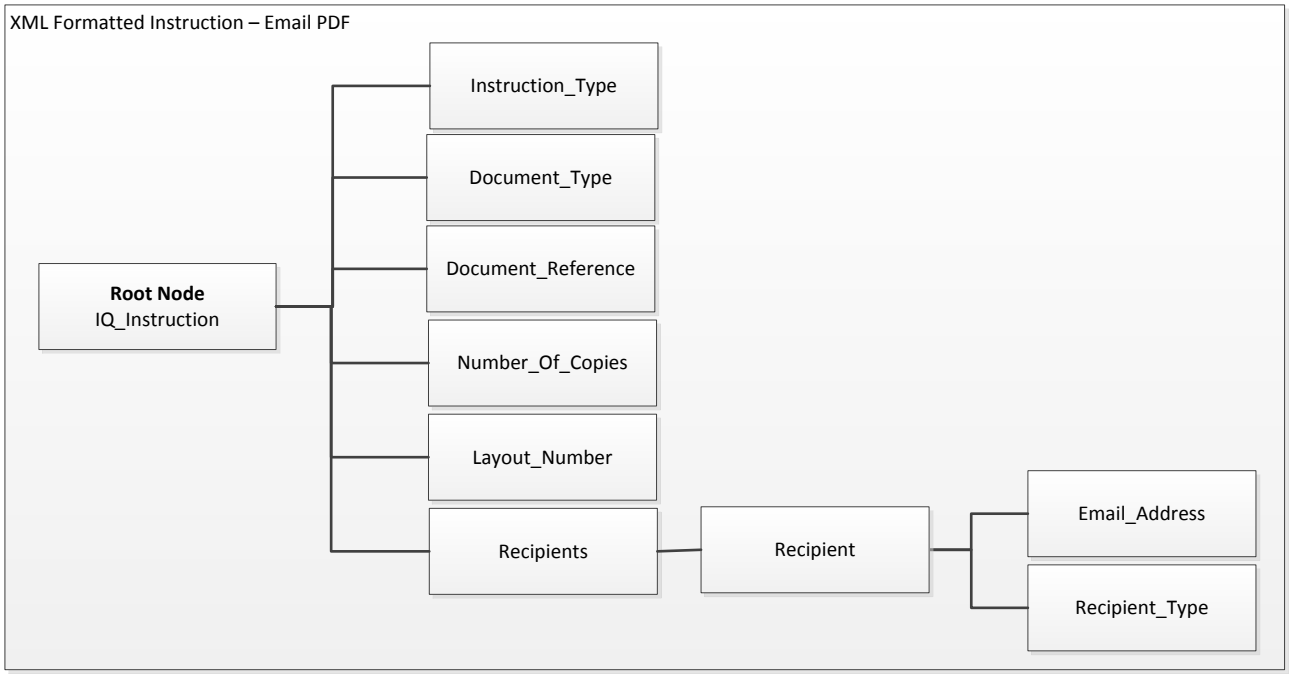
The instruction allows the caller to request the API to

a) Submit the document(s) for emailing purposes after successful submission of the document to the IQ Database. If the document does not pass validation (and is thus not submitted), the failed document(s) will, as a result, also not be submitted for emailing.
b) Submit a request to the API to email a specific document (irrespective of the submitted documents).

   Supported documents types are
   a. Purchase Orders
   b. Sales Orders
   c. Job Cards
   d. Quotes
   e. Invoices
   f. Credit Notes
   g. GRVs
   h. RTSs

The Email instruction allows for the specification of the following parameters (in the form of child XML nodes) in addition to the XML parameters supported by the Print Instruction defined above.

a) Recipients: This node represents a collection of Email Recipients that are receive the email generated via the API call. This node can occur once (1) or many (∞) times – once for each recipient and should consist of the following two subnodes:
   a. Email_Address: This is a string value containing a valid email address. Email address that are not in a valid format (eg. Does not contain the @ sign) will be ignored.
   b. Recipient_Type: This node indicates if the recipient type in terms of i) Normal Recipient ii) Carbon Copy iii) Blind Carbon Copy. See **TIQEnt_API_Email_Recipient_Type** under **Enumerated Type Definitions.**



IQ_API_Instruction_Custom_Report:

Instruction Type: **apiitCustomReport**

This instruction applies to all API Calls and allows the user to submit an instruction to the API to generate a PDF document for emailing purposes.

The Email instruction allows for the specification of the following parameters (in the form of child XML nodes):

a) Report_Name: This represents the report template to be used during generation of the report into PDF format. This corresponds to an FR3 (Fast Reports) report template / layout found within the IQ Retail reports structure.
b) Report_Variables: This node represents a collection of custom data elements to be used a variables within the report writer / designer. The designer of the IQ Report can assume the existence of these variables. The variables can then be sent to the report preview (by the API caller) for interpretation purposes. The Report_Variables node is a collection of Report_Variable (singular) nodes. Each of these nodes must contain the following subnodes:
    a. Variable_Name: This specifies the name of the variable passed to the report preview. It identifies the variable.
    b. Variable_Value: This specifies the value of the variable to be interpreted.

c) Recipients: This node represents a collection of Email Recipients that are receive the email generated via the API call. This node can occur once (1) or many (∞) times – once for each recipient and should consist of the following two subnodes:
    a. Email_Address: This is a string value containing a valid email address. Email address that are not in a valid format (eg. Does not contain the @ sign) will be ignored.
    b. Recipient_Type: This node indicates if the recipient type in terms of i) Normal Recipient ii) Carbon Copy iii) Blind Carbon Copy. See **TIQEnt_API_Email_Recipient_Type** under **Enumerated Type Definitions.**

XML Submissions – Additional Features

The following list of features are additional to the normal submission requirements / features. Each of these have specific applications and apply to specific API Call Types. Each of the calls will be explained within its own section.

IQ Documents - Fallback

This feature allows the caller to request the API to handle failures of document submissions in a different manner. In normal circumstances (without using the fallback option), the API would return an error (and related details) in the event of a document submission failure (due to validation checks).

When using the fallback option, the API will (for supported document types only), ignore all validation errors and post the document in its fallback format instead.

For example: The API caller submits and Invoice (with Sales Order specified as Fallback Type). When the Invoice failes due to (for eg.) a negative stock error, the system will automatically store the document as a Sales Order instead (ignoring the negative stock error).

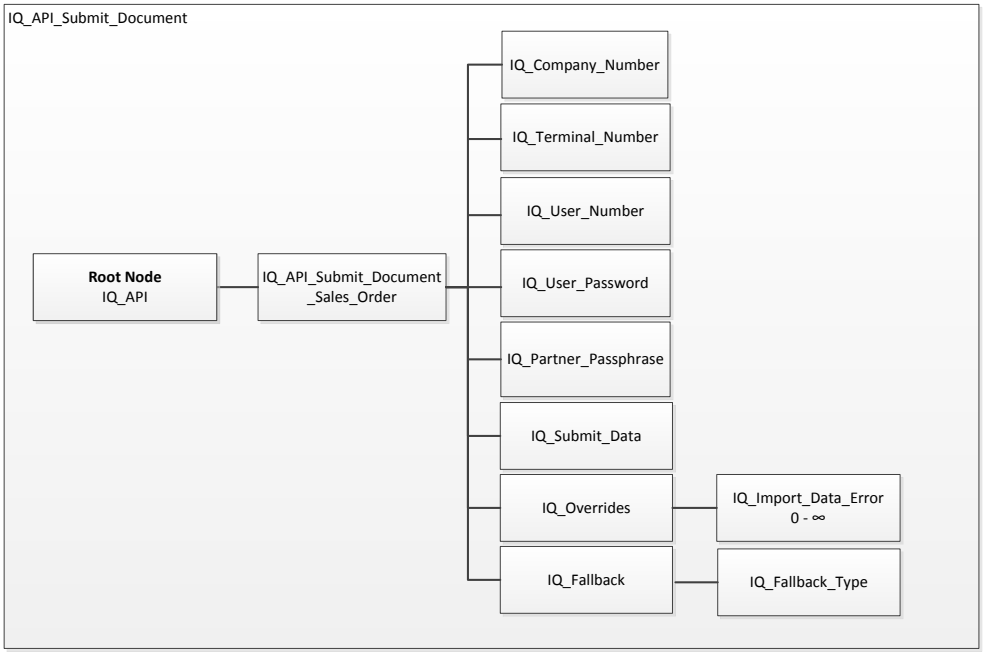
In order to initiate the above feature, the caller must provide an additional node (as a child node of the document root node). The node must be named “IQ_Fallback”, and should contain a child node named “IQ_Fallback_Type”. This latter node must contain an enumerated type (specifying which document type should be used for fallback purposes). The nodes specified here are option (ie. Fallback is an optional feature).

Supported Fallback Types are listed below:

Submission Type	Fallback Type	Fallback Type Node Value
Sales Order	Quote	iqQuote

Invoice	Sales Order	iqSOR
Invoice	Quote	iqQuote

API Call Format:



IQ Documents – Auto Processing

This feature allows the caller to request the API to automatically perform and additional processing function on the successfully submitted document (provided in the original API call). In normal circumstances the API would submit the provided documents and return errors / success data (without the Auto Process option).

When using the Auto Process option, the API will (for supported document types only), try to process the successfully submitted documents into its requested IQ_AutoProcess_Type form (using the data provided in the original document).

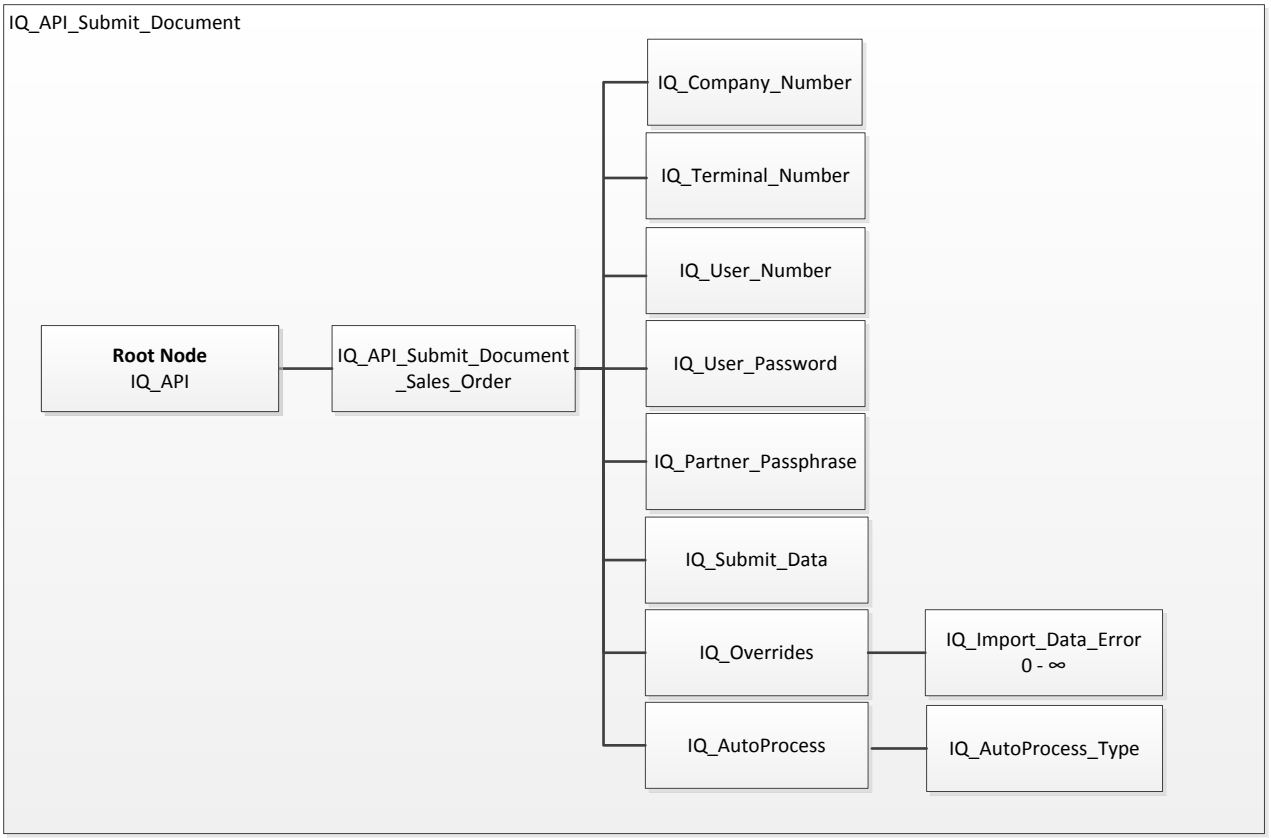
For example: The API Caller submits a Sales Order (with iqINV set as the IQ_AutoProcess_Type). The API will submit the Sales Order and, if successful, also try and convert the Sales Order to an Invoice (following normal IQ practices). In the event of failure, relevant Error Data will be returned.

In order to initiate the above feature, the caller must provide an additional node (as a child node of the document root node). The node must be named “IQ_AutoProcess”, and should contain a child node named “IQ_AutoProcess_Type”. This latter node must contain an enumerated type (specifying which document type should be used for processing purposes). The nodes specified here are option (ie. Auto Processing is an optional feature).

Supported Auto Processing Types are listed below:

Submission Type	Auto Process Type	Fallback Type Node Value
Sales Order	Invoice	iqINV

API Call Format:



Exposed / Available Calls

IQ_API_Test_Int

Description: The method serves as a testing method only. It can be used to determine the successful loading of the IQ API library and successful passing of Integer parameters. This method is a procedure and returns no result. On calling this method, a message dialog will be shown containing the following message (where X is the parameter value that it received from the client application).

This is a Test Message. Call to IQ_API_Test successful with parameter value of X

Type Declaration

[DELPHI] TIQ_EntAPI_Test_Procedure = Procedure(aVal: TIQ_Type_Param_Length); **stdcall**;

[C# Import]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Test_Int")]public static extern void IQ_API_Test_Int(int aValue);
```

Input Parameters:

aVal: TIQ_Type_Param_Length – This is a test parameter of Integer type. This parameter contains the value provided by the host application.

Output Parameters:

None

Example Code [Delphi]:

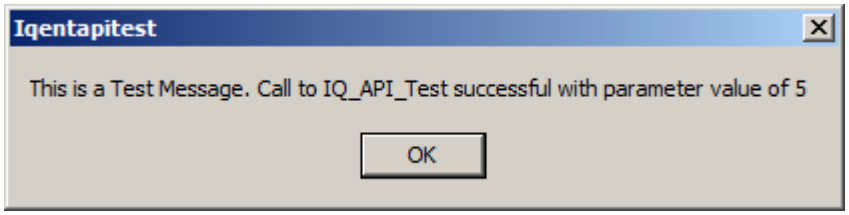
```
procedure TfrmAPITest.btnAPITestClick(Sender: TObject);
var
  FProc: TIQ_EntAPI_Test_Procedure;
begin
  if not LoadDLL then Exit; // FHandle declared outside of this Event
  try
```

```
  FProc := GetProcAddress(FHandle,'IQ_API_Test_Int');
  If Not Assigned(FProc) Then Exit;
  FProc(1);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    [DllImport(@"C:\Somewhere\IQEntAPI.DLL",CallingConvention=
          CallingConvention.StdCall,CharSet=CharSet.Ansi,EntryPoint="IQ_API_Test_Int")]
    public static extern void IQ_API_Test_Int(int aValue);

    private void btnTestInteger_Click(object sender, EventArgs e)
    {
      int FValue = 5;
      IQ_API_Test_Int(FValue);
    }
        //…Other Form Related Code Below
  }
}
```

Output Example:



IQ_API_Test_PChar

Description: This method serves as a test procedure only. It can be used to determine the successful loading of the IQ API and successful passing of string type parameters as PChar. The method accepts a PChar value and its related length. It returns a PChar result and its related length. The method is a procedure and has no result type. Upon calling the method the PChar Parameter and a modified PChar Parameter will be displayed in message dialogs, on screen as received by the IQ API.

Type Declaration

[DELPHI]

```
TIQ_EntAPI_StringTester_Procedure = Procedure(
                    aString              : TIQ_Type_Char_Param;
                    aParam_Length        : TIQ_Type_Param_Length;
                    Out aResult          : TIQ_Type_Char_Param;
                    Var aResultLength    : TIQ_Type_Param_Length);stdcall;
```

[C# Import]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Test_PChar")]public static extern void IQ_API_Test_PChar([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParam_Length, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aString: TIQ_Type_Char_Param - This is a parameter of type PChar and a series of characters.

aParam_Length: TIQ_Type_Param_Length - This parameter contains the length of **aParam**. The method will consider only characters within aParam from the first character up to the length specified in **aParam_Length.**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains a series of characters. Note the keyword **out** – whereby this parameter only returns a result value and accepts no value from the host application.

***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed as a reference parameter (notice the **var** keyword). After a successful call to the method, the value will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult (starting at the first position in the string and extracting characters up to the specified length).

Method Result: Method returns no result.

Example Code [Delphi]:

```
procedure TfrmAPITest.Button1Click(Sender: TObject);
var
  FProc     : TIQ_EntAPI_StringTester_Procedure;
  FFreeProc  : TIQ_EntAPI_Free_PChar;
  FRes      : TIQ_Type_Char_Param;
  FMsg      : TIQ_Type_Char_Param;
  FLength    : TIQ_Type_Param_Length;
  FResLength : TIQ_Type_Param_Length;
begin
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    FProc    := GetProcAddress(FHandle, 'IQ_API_Test_PChar');
    FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

    If Not Assigned(FProc) Then Exit;
    If not Assigned(FFreeProc) Then Exit;

    FResLength := 0;
    FMsg := PChar('Original String :');
    FLength := Length(FMsg);

    FProc(FMsg, FLength, FRes, FResLength);

    ShowMessage('Host Application:' + FRes);
    ShowMessage('Host Application:' + IntToStr(FResLength));

    FFreeProc(FRes);
```

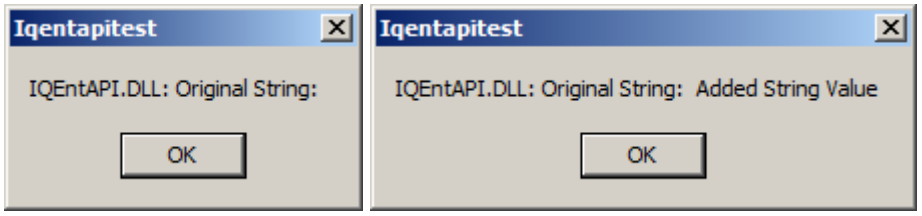
```
 finally
   ReleaseDLL;
 end;
end;
```

Example Code [C#]

```
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {

        [DllImport@"C:\Somewhere\IQEntAPI.DLL,CallingConvention=
    CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Test_PChar")]
    public static extern void IQ_API_Test_PChar([MarshalAs(UnmanagedType.LPStr)]string
    aParam, int aParam_Length, out IntPtr aResult, ref int aResultLength);

     public Form1()
     {
        InitializeComponent();
     }

     private void btnTestPChar_Click(object sender, EventArgs e)
     {
        string FOriginal = "Original String: ";
        IntPtr FResult;
        string FResultString;
        int    FResultLength = 0;

        IQ_API_Test_PChar(FOriginal, FOriginal.Length, out FResult, ref FResultLength);
        FResultString = Marshal.PtrToStringAnsi(FResult);

        MessageBox.Show("Host Application: " + FResultString.Substring(0, FResultLength));
        MessageBox.Show("Host Application: " + FResultLength.ToString());

        IQ_API_Free_PChar(FResult);
     }
```

```
  }
}
```

Output Example:



IQ_API_Free_PChar

Description: ***** See [Memory Allocation] section for reasons why this method has been exposed and should be used in conjunction with any other method that returns a PChar result.**

This method serves as an additional tool for all methods that return a PChar result. All API methods (unless otherwise specified and excluding the Test Methods) return an XML formatted response in a result parameter of type TIQ_Type_Char_Result. In all cases this parameter will be modified by the **out** keyword.

In order to return such value, the IQ API Library needs to allocate memory for the result. It remains the responsibility of the host application to request the IQ API DLL to release such allocated memory after the host application has used the value within it. This method provides serves as a tool to achieve this goal.

Type Declaration

[DELPHI]

```
TIQ_EntAPI_Free_PChar = Procedure (aPChar: TIQ_Type_Char_Param); stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Free_PChar")]public static extern void IQ_API_Free_PChar(IntPtr aChar);
```

Input Paramters:

aPChar: TIQ_Type_Char_Param – This is a PChar containing the value of a result previously returned by another method that allocated memory for its result. The IQ API DLL will free the previously allocated memory for this PChar.

Output Paramters:

None

Example Code: *See relevant sections of methods that require a call to TIQ_EntAPI_Free_PChar (eg. IQ_API_Request_Stock_Attributes)*

IQ_API_Query_Exports

Description: This method serves to provide the host application with all available / exported methods from the IQ API DLL. The host application could potentially use this to gather all available methods for dynamic allocation or to determine if additional exposed methods have been made available.

Type Declaration

[DELPHI]

```
TIQ_EntAPI_DLLQuery_Exports = Procedure(Out aResult: TIQ_Type_Char_Param;   Var aResultLength: TIQ_Type_Param_Length); stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Query_Exports")]public static extern void IQ_API_Query_Exports(out IntPtr aResult, ref int aResultLength);
```

Input Parameters:
None

Output Parameters:

aResult: TIQ_Type_PChar_Param: This is a PChar result containing XML Formatted Data. This data will be found under the IQ_API_Result root node and under the IQ_API_Result_Data node. Note the **out** keyword which specifies that this parameter is intended for output result data only.

aResultLength: TIQ_Type_Param_Length

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Response:

1) IQ_API_Exported_Mehods: This is a collection node containing <Method> subnodes. Each of these subnodes currently contain only the entry point name.

*****See XML Formatted Response section for more details.**

XML Response Example:

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Exported_Methods>
 <Method>IQ_API_Free_PChar</Method>
 <Method>IQ_API_Query_Exports</Method>
 <Method>IQ_API_Request_Creditor_Journal</Method>
 <Method>IQ_API_Request_Creditor_Price_List</Method>
 <Method>IQ_API_Request_Creditor_Store_Departments</Method>
 <Method>IQ_API_Request_Creditor_Store_Departments_Associations</Method>
 <Method>IQ_API_Request_Debtor_Attributes</Method>
 <Method>IQ_API_Request_Debtor_Journal</Method>
 <Method>IQ_API_Request_Debtor_Price_List</Method>
 <Method>IQ_API_Request_Debtor_Store_Departments</Method>
 <Method>IQ_API_Request_Debtor_Store_Departments_Associations</Method>
 <Method>IQ_API_Request_Document_Invoice</Method>
 <Method>IQ_API_Request_Document_Sales_Order</Method>
 <Method>IQ_API_Request_Promotion</Method>
 <Method>IQ_API_Request_SalesRep</Method>
 <Method>IQ_API_Request_Stock_Attributes</Method>
 <Method>IQ_API_Request_Stock_ContractPricing</Method>
 <Method>IQ_API_Submit_Creditor_Journal</Method>
 <Method>IQ_API_Submit_Creditor_Store_Departments</Method>
 <Method>IQ_API_Submit_Debtor_Attributes</Method>
 <Method>IQ_API_Submit_Debtor_Journal</Method>
 <Method>IQ_API_Submit_Debtor_Store_Departments</Method>
```

```
 <Method>IQ_API_Submit_Document_Invoice</Method>
 <Method>IQ_API_Submit_Document_Sales_Order</Method>
 <Method>IQ_API_Submit_Promotion</Method>
 <Method>IQ_API_Submit_SalesRep</Method>
 <Method>IQ_API_Submit_Stock_Attributes</Method>
 <Method>IQ_API_Submit_Stock_ContractPricing</Method>
 <Method>IQ_API_Test_Int</Method>
 <Method>IQ_API_Test_PChar</Method>
</IQ_API_Exported_Methods>
```

Example Code [DELPHI]:

```
procedure TfrmAPITest.Button3Click(Sender: TObject);
var
 FProc: TIQ_EntAPI_DLLQuery_Exports;
 FFreeProc: TIQ_EntAPI_Free_PChar;
 FRes: PChar;
 FResLength: Integer;
begin
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  FProc := GetProcAddress(FHandle,'IQ_API_Query_Exports');
  FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

  If Not Assigned(FProc) Then Exit;
  If not Assigned(FFreeProc) Then Exit;

  FResLength := 0;
  FProc(FRes, FResLength);

  SetResult(Copy(Fres, 1, Freslength));  //Shows result in Memo component
  Label1.Caption := 'Result Length: ' + (IntToStr(FResLength));
  FFreeProc(FRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnGetExports_Click(object sender, EventArgs e)
 {
    IntPtr FResult;
    int FResultLength = 0;
    string FResultString;

    IQ_API_Query_Exports(out FResult, ref FResultLength);
    FResultString = Marshal.PtrToStringAnsi(FResult);

    lstResult.Text = FormatXML(FResultString);
    IQ_API_Free_PChar(FResult);
 }
```

IQ_API_Request_Stock_Attributes

Description: This method allows the client application to request Stock Related attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam              : TIQ_Type_Char_Param;
                                        aParam_Length      : TIQ_Type_Param_Length;
                                        out aResult        : TIQ_Type_Char_Result;
                                        var aResult_Length : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Stock_Attributes")]public static extern int IQ_API_Request_Stock_Attributes([MarshalAs(UnmanagedType.LPStr)]string aParam,
int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

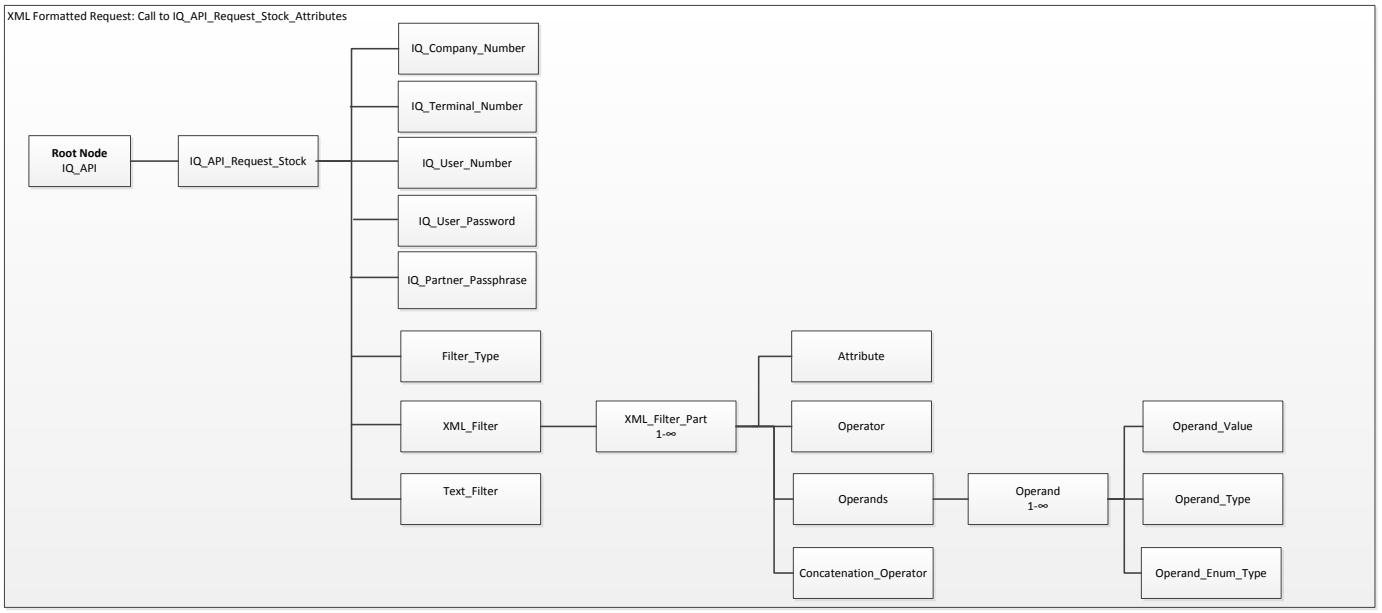
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Stock**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested stock information should be extracted.

Requested_Attributes: This is a group node containing single Attribute nodes. This group can contain between zero (0) and many (∞) Attribute nodes. If there are zero (0) Attribute nodes present, the IQ API will assume that all supported attributes have been requested. Each Attribute node should contain, as its value, the “Friendly” field name as per Stock Master XML Schema (see Appendix A).

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Filter_Type, XML_Filter, Text_Filter: These nodes are used to specify filtering conditions on table data.
***** See [XML Formatted Filters] section for more details on specifying Filter.**

XML Request Example:

Note: This example shows a Stock Request from Company 001, for ALL Stock Attributes (notice that the attributes node have been left blank). The stock records will be filtered and the filter is specified in XML Format (not TEXT format). The XML Filter contains an Operator and Operands (equivalent to a TEXT filter of **Code = 'A'**).

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Stock>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <Requested_Attributes/>
  <Filter_Type>EAPIFilter_XML</Filter_Type>
  <XML_Filter>
   <XML_Filter_Part>
    <Attribute>Stock_Code</Attribute>
    <Operator>opEquals</Operator>
    <Operands>
     <Operand>
      <Operand_Value>A</Operand_Value>
      <Operand_Type>opTypeString</Operand_Type>
     </Operand>
    </Operands>
   </XML_Filter_Part>
  </XML_Filter>
 </IQ_API_Request_Stock>
```

</IQ_API>

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

2) IQ_API_Error
3) IQ_API_Result_Data: An XML formatted result containing the Stock Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQMasterStock** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnAPIStockRequestClick(Sender: TObject);
var
  FProc: TIQ_EntAPI_Request_Procedure; //Request Stock Procedure
  FFreeProc: TIQ_EntAPI_Free_PChar;   //FreePChar Procedure

  FResult: TIQ_Type_Char_Result;      //Result Parameter Value
  FLength: TIQ_Type_Param_Length;     //Result Parameter Length
  FSend: TIQ_Type_Char_Param;         //Request Parameter
  FSendL: TIQ_Type_Param_Length;      //Length or Request Parameter
  FFuncRes: Integer;

  FXML: TNativeXML;
  FNode, FSubNode: TXMLNode;
  FN1, FN2, FN3, FN4 : TXMlNode;
begin
  if not LoadDLL then Exit;         //Handle to DLL Stored in FHandle, declared outside this method
  try
    FProc     := GetProcAddress(FHandle,'IQ_API_Request_Stock_Attributes');
    FFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');
```

```
  if not Assigned(FProc) then Exit;
  if not Assigned(FFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....

   FFuncRes := FProc(FSend,FSendL,FResult,FLength); //Returns error code OR zero if successful

   If FFuncRes = 0 then
   begin
    ShowMessage(Copy(Fresult,1,FLength)); //Contains either Error and / or Result as requested
    ShowMessage(IntToStr(FLength));      //Contains length of result
   End
   Else
    ShowMessage('An Error Occurred. Error Code [' + IntToStr(FFuncRes) + ']')

   FFreeProc(FResult); //Requests the API DLL to Free the allocated memory for FResult
 finally

  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnRequestStock_Click(object sender, EventArgs e)
{
  IntPtr FResult;
  string FResultString;
  int FResultLength;
  string FMessage;
  int FMessageLength;
  int FCallResult;
```

```
    ....
    //Your code that generates the XML formatted request parameter
    //and stores it in FMessage and its length in FMessageLength
    ....

  FResultLength = 0;

  FCallResult = IQ_API_Request_Stock_Attributes(FMessage, FMessageLength, out FResult, ref FResultLength);
  FResultString = Marshal.PtrToStringAnsi(FResult);

  if (FCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
}
```

IQ_API_Request_Debtor_Attributes

Description: This method allows the client application to request Debtor Related attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Request_Debtor_Attributes")]public static extern int IQ_API_Request_Debtor_Attributes([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

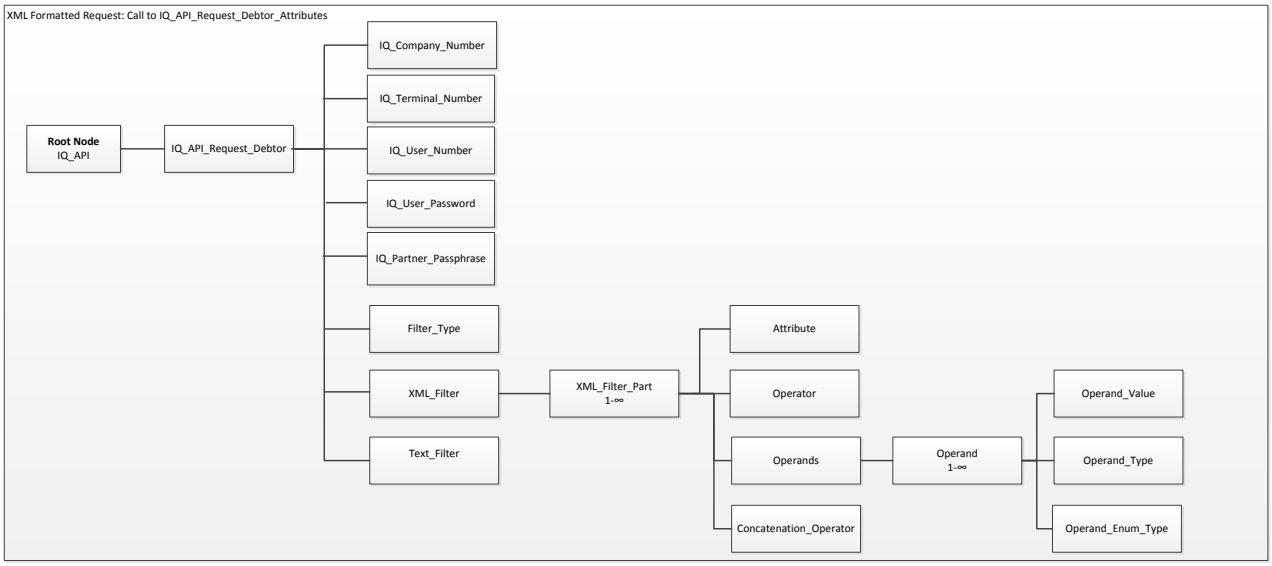
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Debtor**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

Requested_Attributes: This is a group node containing single Attribute nodes. This group can contain between zero (0) and many (∞) Attribute nodes. If there are zero (0) Attribute nodes present, the IQ API will assume that all supported attributes have been requested. Each Attribute node should contain, as its value, the “Friendly” field name as per Debtor Master XML Schema (see Appendix A).

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Filter_Type, XML_Filter, Text_Filter: These nodes are used to specify filtering conditions on table data.
***** See [XML Formatted Filters] section for more details on specifying Filter.**

XML Request Example:

Note: This example shows a Debtor Request from Company 001, for ALL Stock Attributes (notice that the attributes node have been left blank). The debtor records will be filtered and the filter is specified in XML Format (not TEXT format). The XML Filter contains an Operator and Operands (equivalent to a TEXT filter of **Account = 'A'**).

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <Requested_Attributes/>
  <Filter_Type>EAPIFilter_XML</Filter_Type>
  <XML_Filter>
   <XML_Filter_Part>
    <Attribute>Debtor_Account</Attribute>
    <Operator>opEquals</Operator>
    <Operands>
     <Operand>
      <Operand_Value>A</Operand_Value>
      <Operand_Type>opTypeString</Operand_Type>
     </Operand>
    </Operands>
   </XML_Filter_Part>
  </XML_Filter>
 </IQ_API_Request_Debtor>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error

2) IQ_API_Result_Data: An XML formatted result containing the Debtor Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQMasterDebtor** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnAPIDebtorRequestClick(Sender: TObject);
var
  FProc    : TIQ_EntAPI_Request_Procedure; //Request Debtor Procedure
  FFreeProc: TIQ_EntAPI_Free_PChar;       //FreePChar Procedure

  FResult  : TIQ_Type_Char_Result;      //Result Parameter Value
  FLength  : TIQ_Type_Param_Length;     //Result Parameter Length
  FSend    : TIQ_Type_Char_Param;       //Request Parameter
  FSendL   : TIQ_Type_Param_Length;     //Length or Request Parameter
  FFuncRes : Integer;

  FXML: TNativeXML;
  FNode, FSubNode: TXMLNode;
  FN1, FN2, FN3, FN4 : TXMlNode;
begin
  if not LoadDLL then Exit;         //Handle to DLL Stored in FHandle, declared outside this method
  try
    FProc     := GetProcAddress(FHandle,'IQ_API_Request_Debtor_Attributes');
    FFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(FProc) then Exit;
    if not Assigned(FFreeProc) Then Exit;

     ....
    //Your code that generates the XML formatted request parameter
     ....

    FFuncRes := FProc(FSend,FSendL,FResult,FLength); //Returns error code OR zero if successful
```

```
  If FFuncRes = 0 then
  begin
    ShowMessage(Copy(Fresult,1,FLength)); //Contains either Error and / or Result as requested
    ShowMessage(IntToStr(FLength));      //Contains length of result
  End
  Else
    ShowMessage('An Error Occurred. Error Code [' + IntToStr(FFuncRes) + ']'
  FFreeProc(FResult); //Requests the API DLL to Free the allocated memory for FResult
 finally

   ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnRequestDebtor_Click(object sender, EventArgs e)
{
   IntPtr FResult;
   string FResultString;
   int FResultLength;
   string FMessage;
   int FMessageLength;
   int FCallResult;

    ....
    //Your code that generates the XML formatted request parameter
    //and stores it in FMessage and its length in FMessageLength
    ....

  FResultLength = 0;

 FCallResult = IQ_API_Request_Debtor_Attributes(FMessage, FMessageLength, out FResult, ref FResultLength);
 FResultString = Marshal.PtrToStringAnsi(FResult);

 if (FCallResult != 0)
 {
```

```
    MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
}
```

IQ_API_Request_Document_Sales_Order

Description: This method allows the client application to request Sales Order documents (and related items) from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Document_Sales_Order")]public static extern int
IQ_API_Request_Document_Sales_Order([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int
aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of

the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

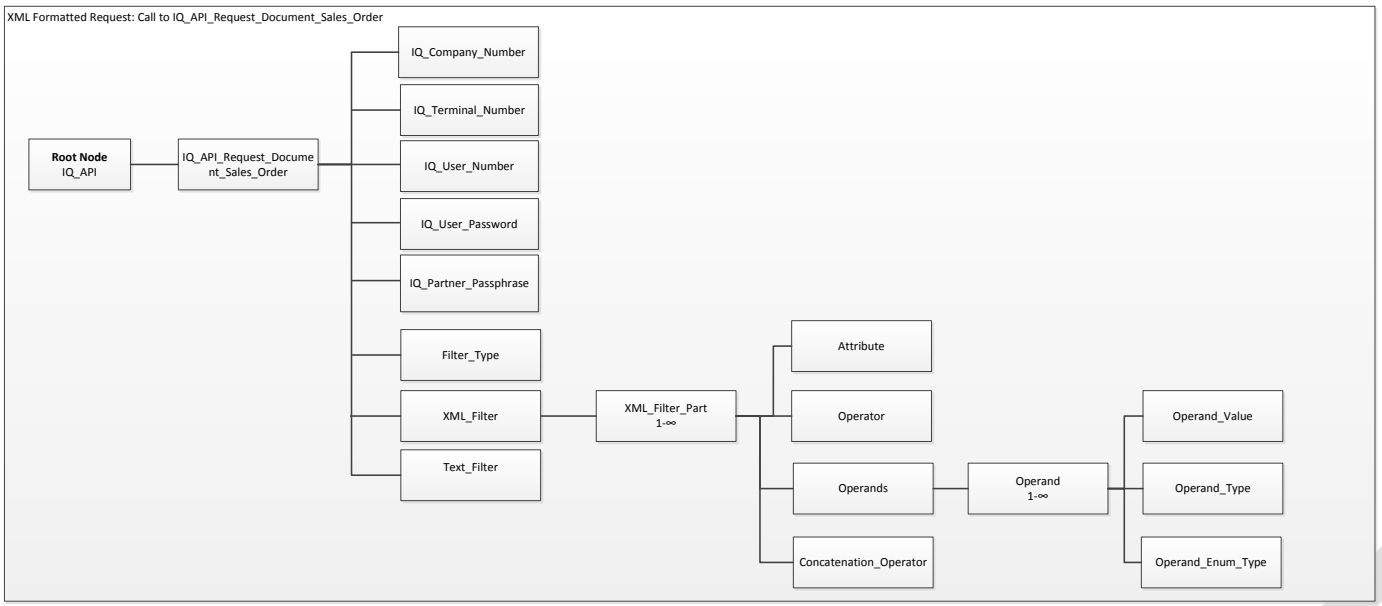
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Document_Sales_Order**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

Requested_Attributes: This request does not support the Attributes feature. In the case of system documents the Attributes for the document are determined by the IQ API DLL.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Filter_Type, XML_Filter, Text_Filter: These nodes are used to specify filtering conditions on table data.
***** See [XML Formatted Filters] section for more details on specifying Filter.**

XML Request Example:
Note: This example shows a Sales Order Request from Company 001. Notice that it contains no Attributes node – the complete Sales Order document will be returned. The Sales Order records will be filtered and the filter is specified in XML Format (not TEXT format). The XML Filter contains an Operator and Operands (equivalent to a TEXT filter of **Document = 'SAL0'**).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API_Request_Document_Sales_Order>
 <IQ_Company_Number>001</IQ_Company_Number>
 <IQ_Terminal_Number>1</IQ_Terminal_Number>
 <IQ_User_Number>1</IQ_User_Number>
 <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
 <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 <Filter_Type>EAPIFilter_XML</Filter_Type>
```

```
 <XML_Filter>
  <XML_Filter_Part>
   <Attribute>Document_Number</Attribute>
   <Operands>
    <Operand>
     <Operand_Value>SAL0</Operand_Value>
     <Operand_Type>opTypeString</Operand_Type>
    </Operand>
   </Operands>
   <Operator>opEquals</Operator>
  </XML_Filter_Part>
 </XML_Filter>
</IQ_API_Request_Document_Sales_Order>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Sales Order records as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQDocumentDebtorSOrder** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnAPISORRequestClick(Sender: TObject);
var
 FProc   : TIQ_EntAPI_Request_Procedure; //Request Sales Order Procedure
 FFreeProc: TIQ_EntAPI_Free_PChar;      //FreePChar Procedure

 FResult  : TIQ_Type_Char_Result;     //Result Parameter Value
 FLength  : TIQ_Type_Param_Length;    //Result Parameter Length
 FSend    : TIQ_Type_Char_Param;      //Request Parameter
 FSendL   : TIQ_Type_Param_Length;    //Length or Request Parameter
 FFuncRes : Integer;

 FXML: TNativeXML;
```

```
 FNode, FSubNode: TXMLNode;
 FN1, FN2, FN3, FN4 : TXMlNode;
begin
 if not LoadDLL then Exit;        //Handle to DLL Stored in FHandle, declared outside this method
 try
  FProc    := GetProcAddress(FHandle,'IQ_API_Request_Document_Sales_Order');
  FFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(FProc) then Exit;
  if not Assigned(FFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....

  FFuncRes := FProc(FSend,FSendL,FResult,FLength); //Returns error code OR zero if successful

  If FFuncRes = 0 then
  begin
   ShowMessage(Copy(Fresult,1,FLength)); //Contains either Error and / or Result as requested
   ShowMessage(IntToStr(FLength));      //Contains length of result
  End
  Else
   ShowMessage('An Error Occurred. Error Code [' + IntToStr(FFuncRes) + ']')

  FFreeProc(FResult); //Requests the API DLL to Free the allocated memory for FResult
 finally

  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnRequestSOR_Click(object sender, EventArgs e)
 {
    IntPtr FResult;
```

```
    string FResultString;
    int FResultLength;
    string FMessage;
    int FMessageLength;
    int FCallResult;

            ....
            //Your code that generates the XML formatted request parameter
            ....

    FMessage = FStringWriter.ToString();
    FMessageLength = FMessage.Length;
    FResultLength = 0;

            FCallResult = IQ_API_Request_Document_Sales_Order(FMessage, FMessageLength, out FResult, ref    FResultLength);

    FResultString = Marshal.PtrToStringAnsi(FResult);

    if (FCallResult != 0)
    {
       MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
    }
    lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
}
```

IQ_API_Request_Document_Purchase_Order

Description: This method allows the client application to request Purchase Order documents (and related items) from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Document_Purchase_Order")]public static extern int
IQ_API_Request_Document_Purchase_Order([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int
aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

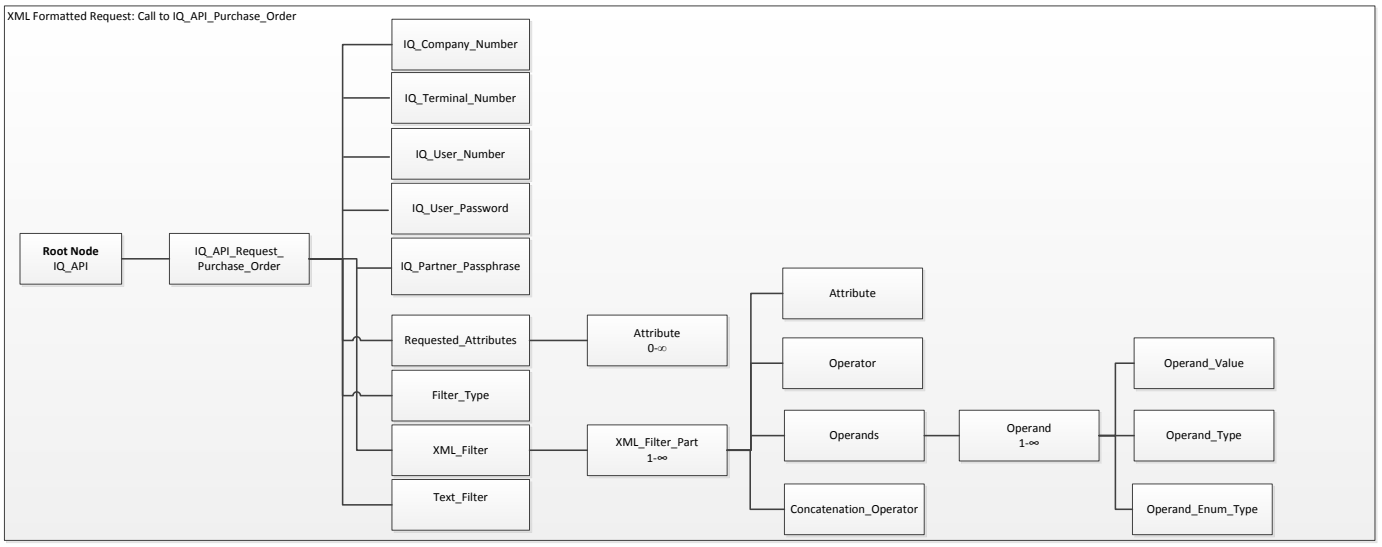
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Document_Purchase_Order**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

Requested_Attributes: This request does not support the Attributes feature. In the case of system documents the Attributes for the document are determined by the IQ API DLL.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Filter_Type, XML_Filter, Text_Filter: These nodes are used to specify filtering conditions on table data.
***** See [XML Formatted Filters] section for more details on specifying Filter.**

XML Request Example:

Note: This example shows a Purchase Order Request from Company 001. Notice that it contains no Attributes node – the complete Purchase Order document will be returned. The Purchase Order records will be filtered and the filter is specified in XML Format (not TEXT format). The XML Filter contains an Operator and Operands (equivalent to a TEXT filter of **Document = 'POR0'**).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API_Request_Document_Purchase_Order>
 <IQ_Company_Number>001</IQ_Company_Number>
 <IQ_Terminal_Number>1</IQ_Terminal_Number>
 <IQ_User_Number>1</IQ_User_Number>
 <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
 <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 <Filter_Type>EAPIFilter_XML</Filter_Type>
 <XML_Filter>
  <XML_Filter_Part>
   <Attribute>Document_Number</Attribute>
   <Operands>
    <Operand>
     <Operand_Value>POR0</Operand_Value>
     <Operand_Type>opTypeString</Operand_Type>
    </Operand>
   </Operands>
   <Operator>opEquals</Operator>
  </XML_Filter_Part>
 </XML_Filter>
</IQ_API_Request_Document_Sales_Order>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error

2) IQ_API_Result_Data: An XML formatted result containing the Purchase Order records as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQDocumentCreditorPOrder** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnAPIPORRequestClick(Sender: TObject);
var
  FProc    : TIQ_EntAPI_Request_Procedure; //Request Purchase Order Procedure
  FFreeProc: TIQ_EntAPI_Free_PChar;       //FreePChar Procedure

  FResult  : TIQ_Type_Char_Result;      //Result Parameter Value
  FLength  : TIQ_Type_Param_Length;     //Result Parameter Length
  FSend    : TIQ_Type_Char_Param;       //Request Parameter
  FSendL   : TIQ_Type_Param_Length;     //Length or Request Parameter
  FFuncRes : Integer;

  FXML: TNativeXML;
  FNode, FSubNode: TXMLNode;
  FN1, FN2, FN3, FN4 : TXMlNode;
begin
  if not LoadDLL then Exit;         //Handle to DLL Stored in FHandle, declared outside this method
  try
    FProc    := GetProcAddress(FHandle,'IQ_API_Request_Document_Purchase_Order');
    FFreeProc := GetProcAddress(FHandle,'IQ_API_Free_PChar');

    if not Assigned(FProc) then Exit;
    if not Assigned(FFreeProc) Then Exit;

     ....
    //Your code that generates the XML formatted request parameter
     ....

    FFuncRes := FProc(FSend,FSendL,FResult,FLength); //Returns error code OR zero if successful

    If FFuncRes = 0 then
```

```
  begin
    ShowMessage(Copy(Fresult,1,FLength)); //Contains either Error and / or Result as requested
    ShowMessage(IntToStr(FLength));      //Contains length of result
  End
  Else
    ShowMessage('An Error Occurred. Error Code [' + IntToStr(FFuncRes) + ']')

  FFreeProc(FResult); //Requests the API DLL to Free the allocated memory for FResult
finally

  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnRequestPOR_Click(object sender, EventArgs e)
 {
    IntPtr FResult;
    string FResultString;
    int FResultLength;
    string FMessage;
    int FMessageLength;
    int FCallResult;

            ....
            //Your code that generates the XML formatted request parameter
            ....

    FMessage = FStringWriter.ToString();
    FMessageLength = FMessage.Length;
    FResultLength = 0;

          FCallResult = IQ_API_Request_Document_Purchase_Order(FMessage, FMessageLength, out FResult, ref   FResultLength);

    FResultString = Marshal.PtrToStringAnsi(FResult);
```

```
    if (FCallResult != 0)
    {
       MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
    }
    lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
}
```

IQ_API_Request_Document_Invoice

Description: This method allows the client application to request Invoice documents (and related items) from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Document_Invoice")]public static extern int IQ_API_Request_Document_Invoice([MarshalAs(UnmanagedType.LPStr)]string
aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

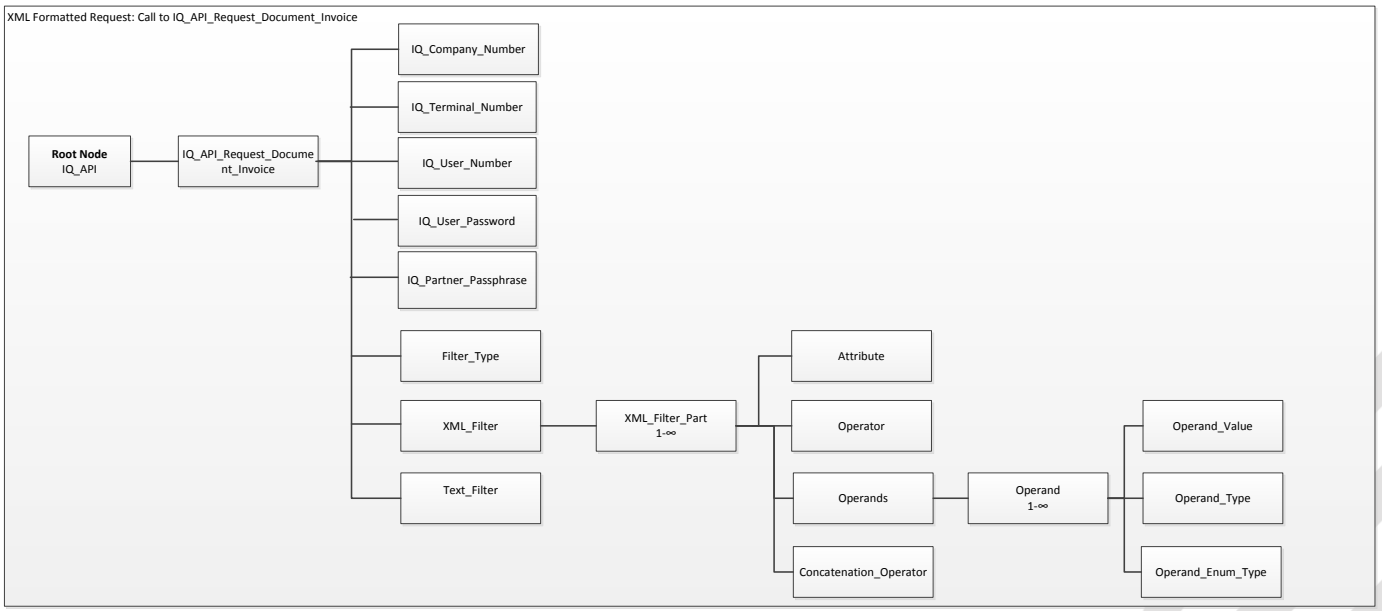
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Document_Invoice**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

Requested_Attributes: This request does not support the Attributes feature. In the case of system documents the Attributes for the document are determined by the IQ API DLL.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Filter_Type, XML_Filter, Text_Filter: These nodes are used to specify filtering conditions on table data.
***** See [XML Formatted Filters] section for more details on specifying Filter.**

XML Request Example:
Note: This example shows a Invoice Request from Company 001. Notice that it contains no Attributes node – the complete Sales Order document will be returned. The Invoice records will be filtered and the filter is specified in XML Format (not TEXT format). The XML Filter contains an Operator and Operands (equivalent to a TEXT filter of **Document = 'INV105'**).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API_Request_Document_Invoice>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
```

```
<IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
<IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
<Filter_Type>EAPIFilter_XML</Filter_Type>
<XML_Filter>
 <XML_Filter_Part>
  <Attribute>Document_Number</Attribute>
  <Operands>
   <Operand>
    <Operand_Value>INV105</Operand_Value>
    <Operand_Type>opTypeString</Operand_Type>
   </Operand>
  </Operands>
  <Operator>opEquals</Operator>
 </XML_Filter_Part>
</XML_Filter>
</IQ_API_Request_Document_Invoice>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Invoice records as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQDocumentDebtorInvoice** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.Button4Click(Sender: TObject);
var
 FProc    : TIQ_EntAPI_Request_Procedure;
 FFreeProc: TIQ_EntAPI_Free_PChar;

 FResult: TIQ_Type_Char_Result;
 FLength: TIQ_Type_Param_Length;
 FSend: TIQ_Type_Char_Param;
```

```
 FSendL: Integer;
 FXML: TNativeXML;
 FNode: TXMLNode;
 FSubNode: TXMLNode;
 FN1, FN2, FN3: TXMLNode;
begin
 if not LoadDLL then Exit;
 try
  FProc := GetprocAddress(FHandle,'IQ_API_Request_Document_Invoice');
  FFreeProc := GetprocAddress(FHandle,'IQ_API_Free_PChar');

  If Not Assigned(FProc) Then Exit;
  If Not Assigned(FFreeProc) Then Exit;

  FXML := TNativeXML.Create;
  try
   FXML.Root.Name := 'IQ_API';
   FNode := FXML.Root.NodeNew('IQ_API_Request_Document_Invoice');
   FNode.NodeNew('IQ_Company_Number').ValueAsString := '001';
   FNode.NodeNew('Requested_Attributes');
   FNode.NodeNew('Filter_Type').ValueAsString := 'EAPIFilter_XML';

   FSubNode := FNode.NodeNew('XML_Filter');
   FN1 := FSubNode.NodeNew('XML_Filter_Part');
   FN2 := FN1.NodeNew('Attribute');
   FN2.ValueAsString := 'Document_Number';

   FN2 := FN1.NodeNew('Operator');
   FN2.ValueAsString := 'opEquals';

   FN2 := FN1.NodeNew('Operands').NodeNew('Operand');
   FN3 := fn2.NodeNew('Operand_Value');
   FN3.ValueAsString := 'INV105';

   FN3 := FN2.NodeNew('Operand_Type');
   FN3.ValueAsString := 'opTypeString';
```

```
//    FNode.NodeNew('Text_Filter').ValueAsString := 'CODE = ''A''';

    FSend := PChar(FXML.Root.WriteToString);
    FSendL := Length(FSend);

    FProc(FSend,FSendL,FResult,FLength);
    SetResult(Copy(FResult, 1, FLength));

    FFreeProc(FResult);
   finally
    FreeAndNil(FXML);
   end;
  finally
   ReleaseDLL;
  end;

end;
```

Example Code [C#]:

```
private void button1_Click(object sender, EventArgs e)
{
   IntPtr FResult;
   string FResultString;
   int FResultLength;
   string FMessage;
   int FMessageLength;
   int FCallResult;

   StringWriter FStringWriter = new StringWriter();

   using (XmlWriter FWriter = XmlWriter.Create(FStringWriter))
   {
      FWriter.WriteStartDocument();
      FWriter.WriteStartElement("IQ_API");
      FWriter.WriteStartElement("IQ_API_Request_Document_Invoice");
```

```
    FWriter.WriteElementString("IQ_Company_Number", "001");
    FWriter.WriteElementString("Filter_Type", "EAPIFilter_XML");
    FWriter.WriteStartElement("XML_Filter");
    FWriter.WriteStartElement("XML_Filter_Part");
    FWriter.WriteElementString("Attribute", "Document_Number");
    FWriter.WriteStartElement("Operands");
    FWriter.WriteStartElement("Operand");
    FWriter.WriteElementString("Operand_Value", "INV108");
    FWriter.WriteElementString("Operand_Type", "opTypeString");
    FWriter.WriteEndElement(); //Operand
    FWriter.WriteEndElement(); //Operands
    FWriter.WriteElementString("Operator", "opEquals");
    FWriter.WriteEndElement(); //XML_Filter_APart
    FWriter.WriteEndElement(); //XMLFilter
    FWriter.WriteEndElement(); //IQ_API_Request_Stock
    FWriter.WriteEndElement(); //IQ_API
    FWriter.WriteEndDocument();
    FWriter.Flush();
  }

  FMessage = FStringWriter.ToString();
  FMessageLength = FMessage.Length;
  FResultLength = 0;

  FCallResult = IQ_API_Request_Document_Invoice(FMessage, FMessageLength, out FResult, ref FResultLength);

  FResultString = Marshal.PtrToStringAnsi(FResult);

  if (FCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
  }
  lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
}
```

IQ_API_Request_Debtor_Journal

Description: This method allows the client application to request Debtor Journal Related attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length        : TIQ_Type_Param_Length;
                                        out aResult          : TIQ_Type_Char_Result;
                                        var aResult_Length   : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Debtor_Journal")]
    public static extern int IQ_API_Request_Debtor_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

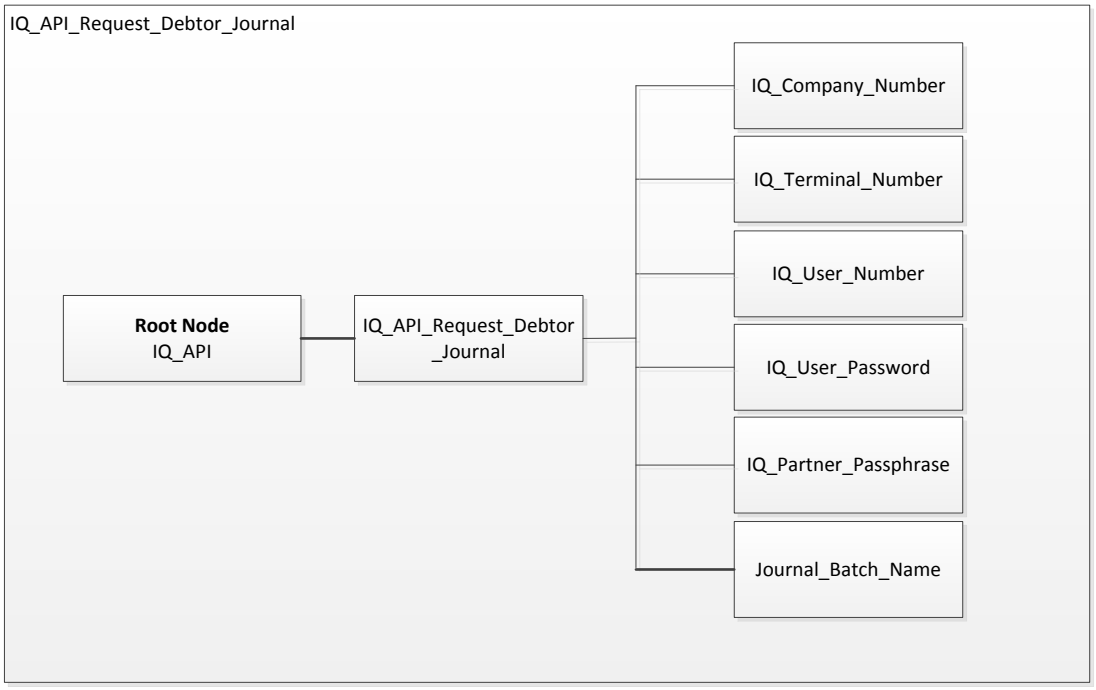
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Debtor_Journal**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Journal_Batch_Name: This value is optional and represents the name of a stored / saved batch within the IQ Retail system. If the batch exists, the data will be returned. If this node is omitted, the API will return the batch data currently active for the emulated terminal (as specified in the IQ_Terminal_Number node).

XML Request Example:
Note: This example shows a Debtor Journal Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Debtor_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Debtor Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQJournalDebtor** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDJournalRequestClick(Sender: TObject);
var
  LProc     : TIQ_EntAPI_Request_Procedure;  //Request Debtor Procedure
  LFreeProc : TIQ_EntAPI_Free_PChar;                    //FreePChar Procedure
  LResult   : TIQ_Type_Char_Result;                     //Result Parameter Value
  LLength   : TIQ_Type_Param_Length;                    //Result Parameter Length
  LSend     : TIQ_Type_Char_Param;                      //Request Parameter
  LXml      : TNativeXml;
  LNode     : TXmlNode;
  LSendL    : Integer;                                  //Length or Request Parameter
  LPassword : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
  try
    LProc     := GetProcAddress(FHandle, 'IQ_API_Request_Debtor_Journal');
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) Then Exit;

      ....
      //Your code that generates the XML formatted request parameter
      ....
  finally
    ReleaseDLL;
  end;
end;
```

Example Code [C#]:

```
private void btnDJournalRequest_Click(object sender, EventArgs e)
{
  IntPtr LResult;
```

```
string LResultString;
int LResultLength;
string LMessage;
int LMessageLength;
int LCallResult;

  ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
LResultLength  = 0;

LCallResult = IQ_API_Request_Debtor_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Creditor_Journal

Description: This method allows the client application to request Creditor Journal Related attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]
TIQ_EntAPI_Request_Procedure = Function(aParam : TIQ_Type_Char_Param;

```
                                    aParam_Length       : TIQ_Type_Param_Length;
                                    out aResult         : TIQ_Type_Char_Result;
                                    var aResult_Length  : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Creditor_Journal")]
    public static extern int IQ_API_Request_Debtor_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

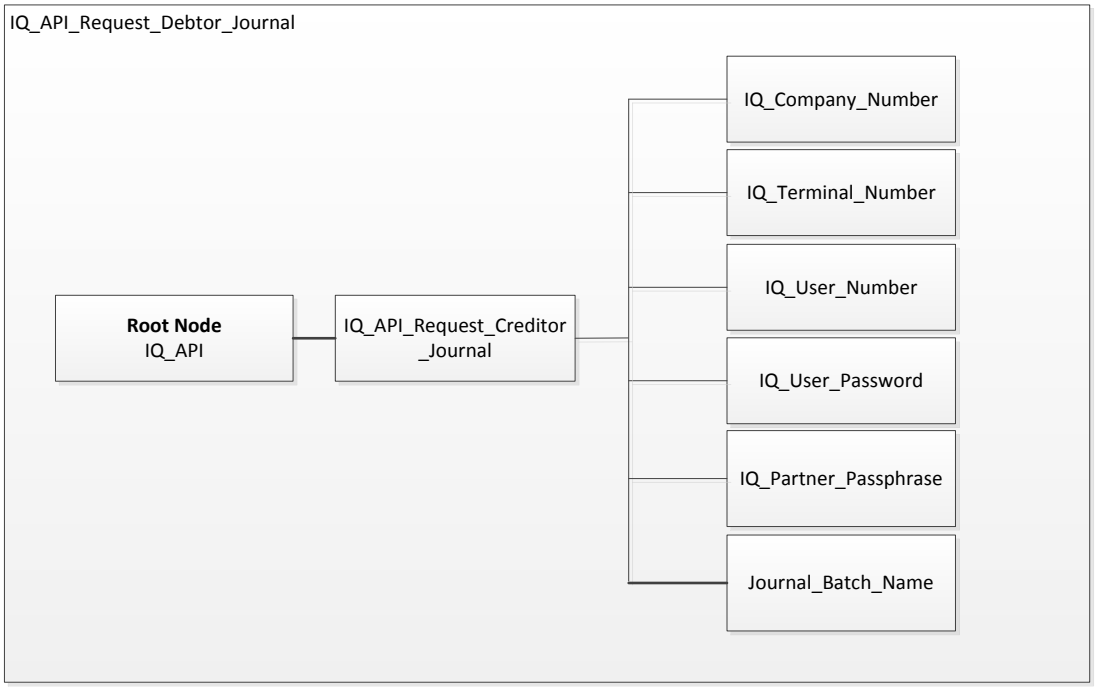
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Creditor_Journal**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Journal_Batch_Name: This value is optional and represents the name of a stored / saved batch within the IQ Retail

system. If the batch exists, the data will be returned. If this node is omitted, the API will return the batch data currently active for the emulated terminal (as specified in the IQ_Terminal_Number node).

XML Request Example:

Note: This example shows a Debtor Journal Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Creditor_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Creditor_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Debtor Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQJournalCreditor** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnCJournalRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Creditor Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
```

```
 LXml     : TNativeXml;
 LNode    : TXmlNode;
 LSendL   : Integer;                                  //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Creditor_Journal');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

    ....
    //Your code that generates the XML formatted request parameter
    ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCJournalRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
```

```
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Creditor_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Ledger_Journal

Description: This method allows the client application to request Ledger Journal Related attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Ledger_Journal")]
    public static extern int IQ_API_Request_Ledger_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

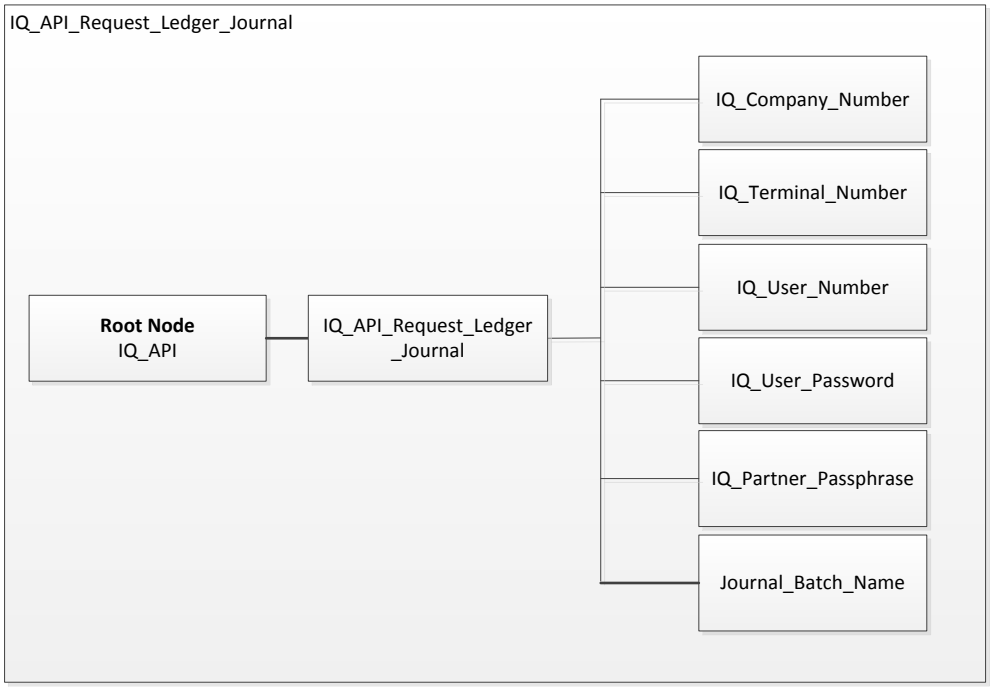
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Ledger_Journal**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

Journal_Batch_Name: This value is optional and represents the name of a stored / saved batch within the IQ Retail

system. If the batch exists, the data will be returned. If this node is omitted, the API will return the batch data currently active for the emulated terminal (as specified in the IQ_Terminal_Number node).

XML Request Example:

Note: This example shows a Debtor Journal Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Ledger_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Creditor_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Ledger Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQJournalLedger** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnLJournalRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Ledger Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
```

```
 LXml     : TNativeXml;
 LNode    : TXmlNode;
 LSendL   : Integer;                                  //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Ledger_Journal');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

    ....
    //Your code that generates the XML formatted request parameter
    ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnLJournalRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
```

```
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Ledger_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Debtor_Store_Departments

Description: This method allows the client application to request Debtor Store Department attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam              : TIQ_Type_Char_Param;
                                        aParam_Length      : TIQ_Type_Param_Length;
                                        out aResult        : TIQ_Type_Char_Result;
                                        var aResult_Length : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Debtor_Store_Departments")]
    public static extern int IQ_API_Request_Debtor_Store_Departments ([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out
IntPtr aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

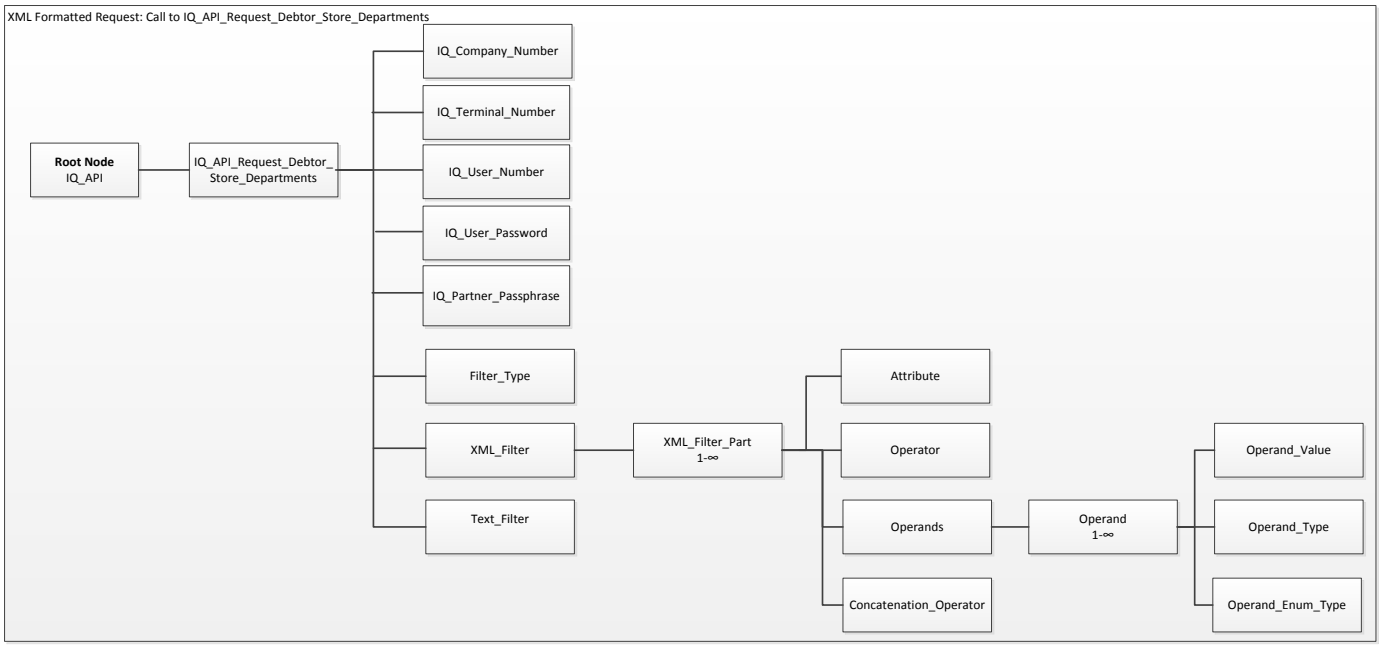
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Debtor_Store_Department.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:

Note: This example shows a Debtor Store Department Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor_Store_Departments>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Debtor_Store_Departments >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Debtor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQDebtorStoreDepartmentsMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDStoreDeptRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                              //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
```

```
if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_ IQ_API_Request_Debtor_Store_Departments ');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....
finally
  ReleaseDLL;
end;
end;
```

Example Code [C#]:

```
private void btnDStoreDeptRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

  ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
  ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Debtor_Store_Department(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
```

```
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Creditor_Store_Departments

Description: This method allows the client application to request Creditor Store Department attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Creditor_Store_Departments")]
    public static extern int IQ_API_Request_Creditor_Store_Departments ([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength,
out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

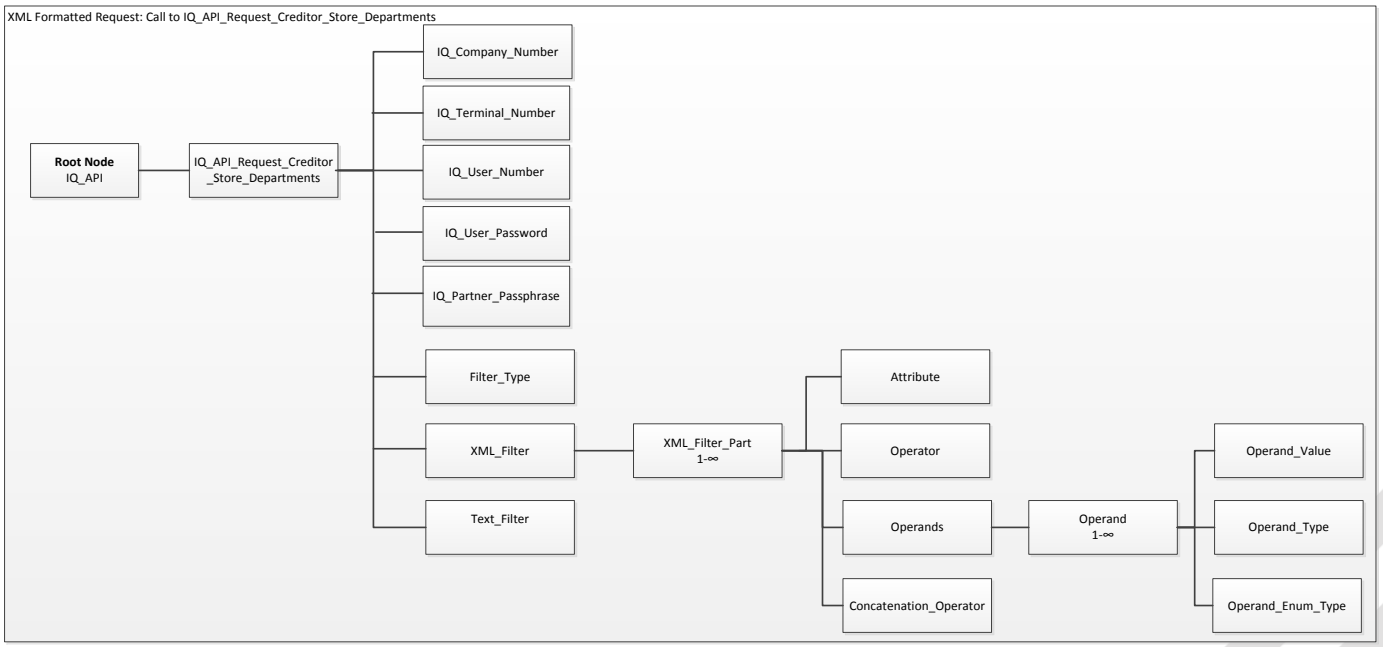
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Creditor_Store_Department.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:
Note: This example shows a Creditor Store Department Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Creditor_Store_Departments>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Creditor_Store_Departments >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error

2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQCreditorStoreDepartmentsMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnCStoreDeptRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                  //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                   //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                  //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                    //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                                //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_ IQ_API_Request_Creditor_Store_Departments ');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCStoreDeptRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Creditor_Store_Department(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Stock_ContractPricing

Description: This method allows the client application to request Stock Contract Pricing attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "
IQ_API_Request_Stock_ContractPricing")]
    public static extern int IQ_API_Request_Stock_ContractPricing ([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

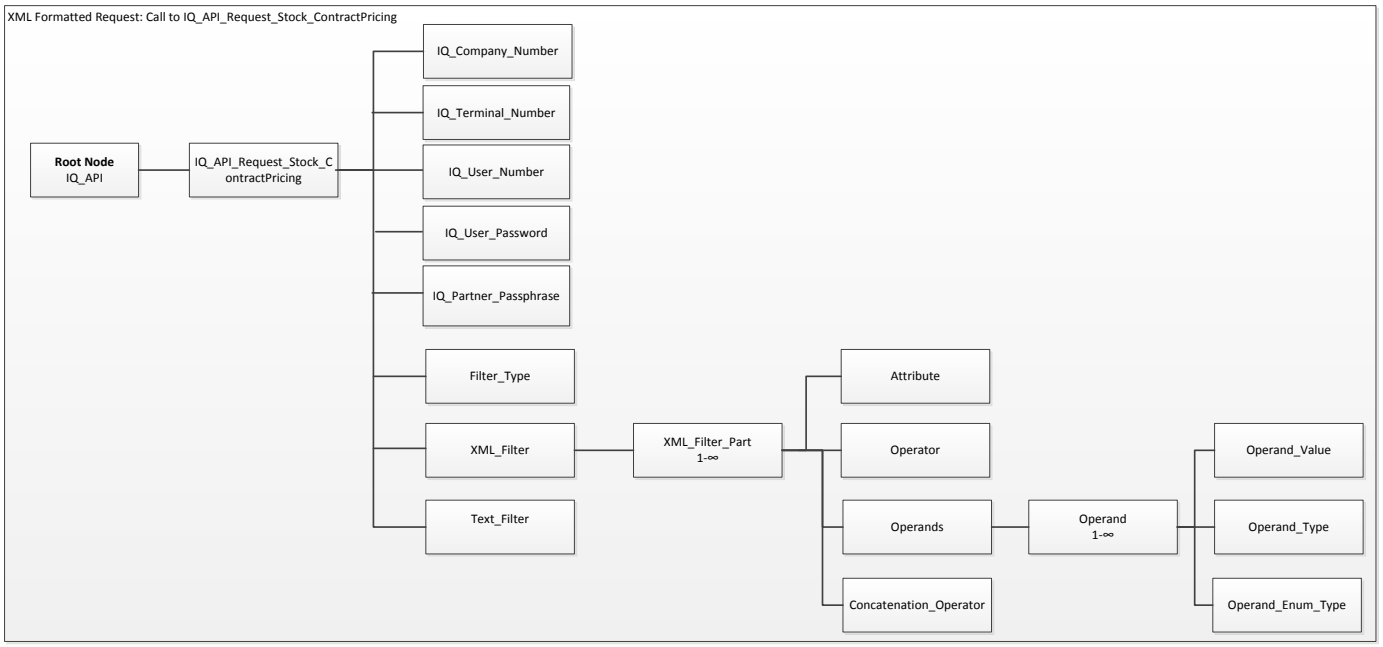
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Stock_ContractPricing.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:

Note: This example shows a Stock Contract Pricing Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Stock_ContractPricing>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Stock_ContractPricing>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQStockContractPriceMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnStockContractRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;              //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;               //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;              //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                //Request Parameter
 LXml      : TNativeXml;
```

```
 LNode    : TXmlNode;
 LSendL   : Integer;                                  //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Stock_ContractPricing');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnStockContractRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

  ....
  //Your code that generates the XML formatted request parameter
  //and stores it in LMessage and its length in FMessageLength
  ....
 LResultLength  = 0;
```

```
LCallResult = IQ_API_Request_Stock_ContractPricing (LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Stock_ContractPricing_Itemized

Description: This method allows the client application to request Stock Pricing Contract attributes and records from the IQ application / related database tables. This method differs from *IQ_API_Request_Stock_ContractPricing* in the sense that all data is itemized by the IQ API engine. All references to groups of stock items (via departments and sub departments) or groups of accounts (via debtor groups) will be analyzed and the collection of data will be returned with each record representing a single item within the groupings (as mentioned above).

This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Stock_ContractPricing_Itemized")]
public static extern int IQ_API_Request_Stock_ContractPricing_Itemized([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out
IntPtr aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

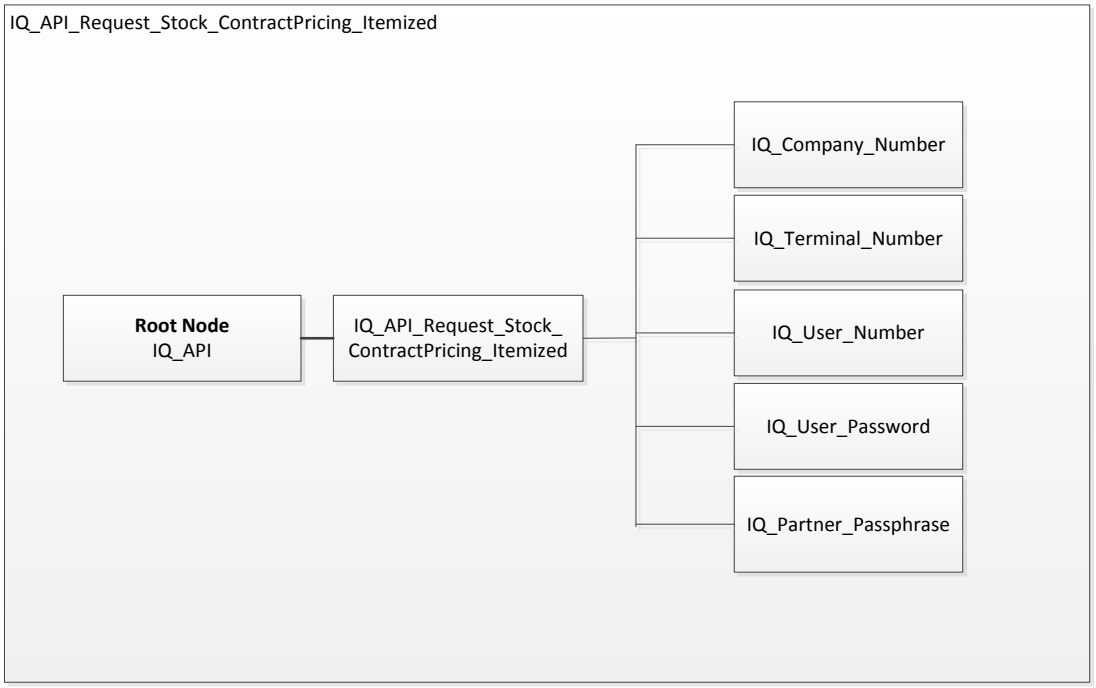
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Stock_ContractPricing_Itemized**.

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested creditor price list information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal within the IQ Software from which the API is requesting information.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user within the IQ Software from which the API is requesting information.

IQ_User_Password: This is the password of the IQ User within the IQ Software. The password sent to the API needs to be hashed as well as converted to hex.

XML Request Example:
Note: This example shows a Contract Price Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Stock_ContractPricing_Itemized>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
 </IQ_API_Request_Stock_ContractPricing_Itemized >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Itemized Contract Price Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQStockContractPriceItemized** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnStockContractPricingItemizedRequestClick(Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Creditor Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                              //Length or Request Parameter
 LPassword : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Stock_ContractPricing_Itemized');
```

```
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

    ....
    //Your code that generates the XML formatted request parameter
    ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnStockContractPricingItemizedRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Stock_ContractPricing_Itemized(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }
```

```
lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Promotion

Description: This method allows the client application to request Stock Promotion attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Promotion")]
    public static extern int IQ_API_Request_Promotion ([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult,
ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

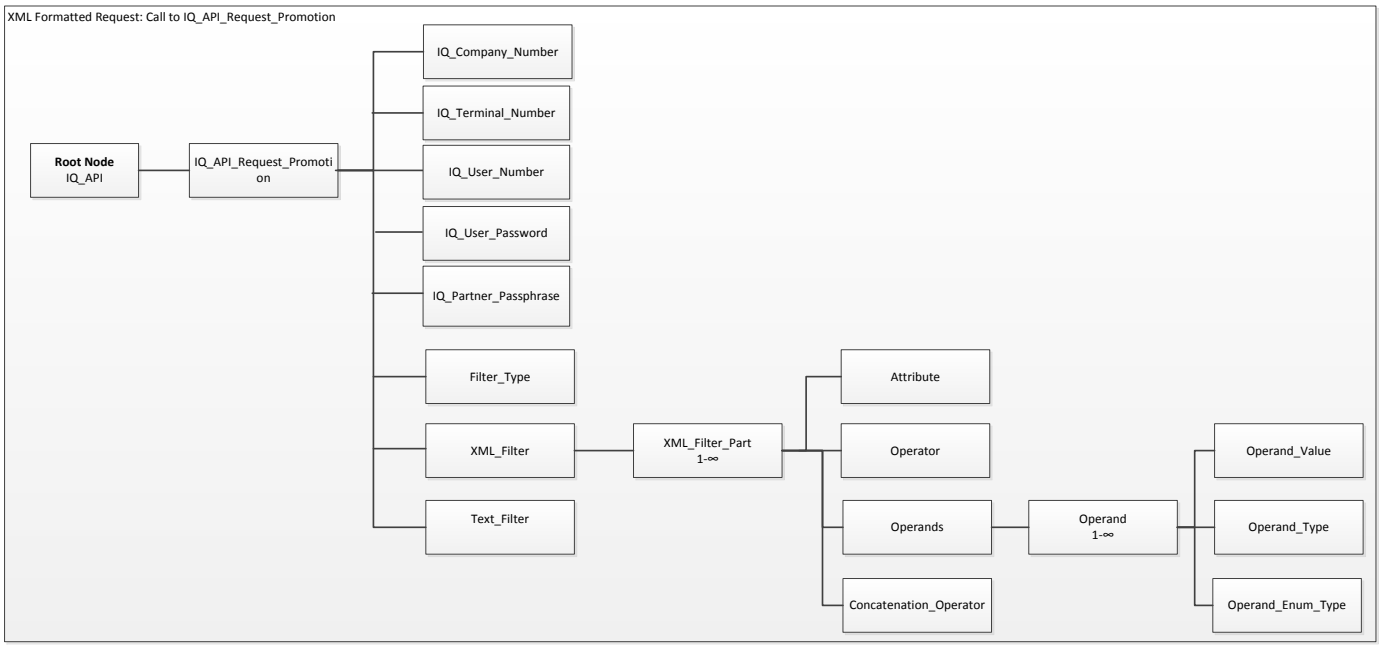
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Promotion.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:
Note: This example shows a Stock Promotion Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Promotion>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Promotion >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQPromotionsMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnPromotionsRequestClick (Sender: TObject);
var
  LProc     : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
  LFreeProc : TIQ_EntAPI_Free_PChar;                    //FreePChar Procedure
  LResult   : TIQ_Type_Char_Result;                     //Result Parameter Value
  LLength   : TIQ_Type_Param_Length;                    //Result Parameter Length
  LSend     : TIQ_Type_Char_Param;                      //Request Parameter
  LXml      : TNativeXml;
  LNode     : TXmlNode;
  LSendL    : Integer;                                  //Length or Request Parameter
  LPassword : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
  try
    LProc     := GetProcAddress(FHandle, 'IQ_API_Request_Promotion');
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) Then Exit;

      ....
      //Your code that generates the XML formatted request parameter
      ....
  finally
    ReleaseDLL;
  end;
end;
```

Example Code [C#]:

```
private void btnPromotionsRequest_Click(object sender, EventArgs e)
{
  IntPtr LResult;
  string LResultString;
  int LResultLength;
```

```
string LMessage;
int LMessageLength;
int LCallResult;

  ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
  ....
LResultLength  = 0;

LCallResult = IQ_API_Request_Promotion(LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Sales_Rep

Description: This method allows the client application to request Sales Representative / Area Manager attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Request_Sales_Rep")]

    public static extern int IQ_API_Request_Sales_Rep ([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

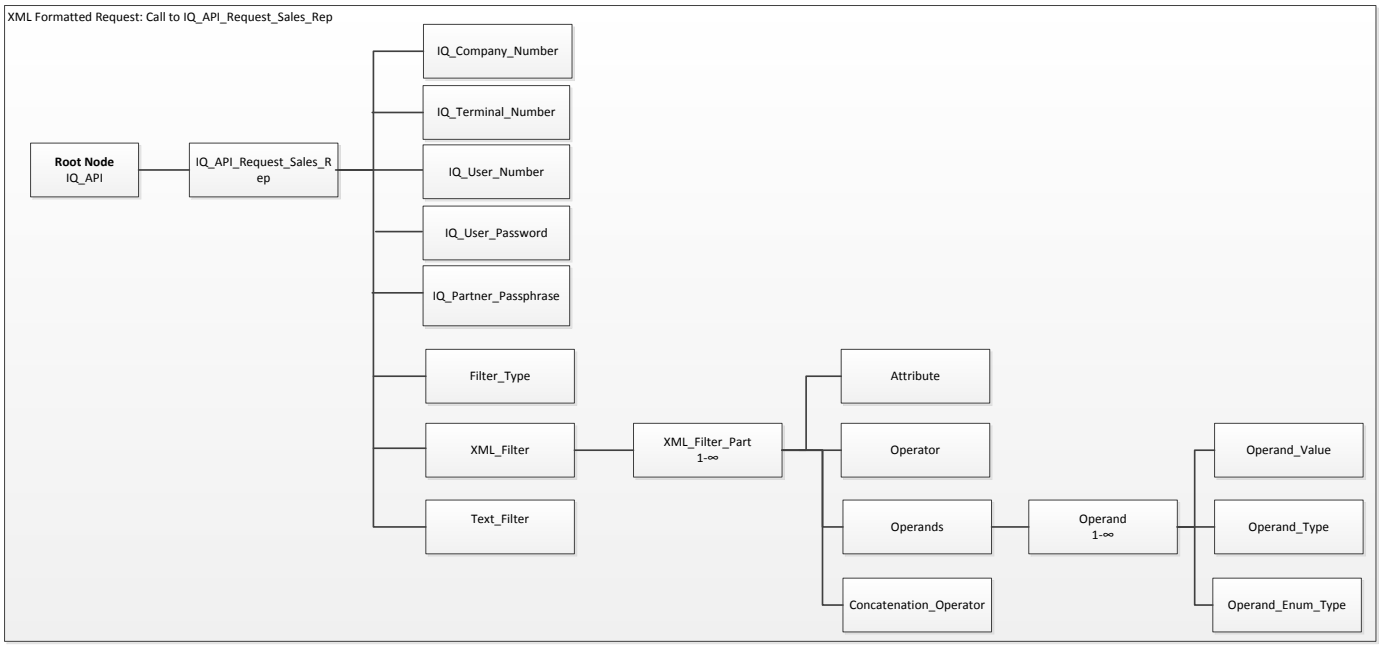
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Sales_Rep.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:

Note: This example shows a Sales Representative Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Sales_Rep>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Sales_Rep>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQRepresentativesMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnSaleRepRequestClick (Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                              //Length or Request Parameter
 LPassword : String;
begin
```

```
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Sales_Rep');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnSalesRepRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

  ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Sales_Rep (LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);
```

```
 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Debtor_Price_List

Description: This method allows the client application to request Debtor Price List(s) attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Debtor_Price_List")]
    public static extern int IQ_API_Request_Debtor_Price_List([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of

the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

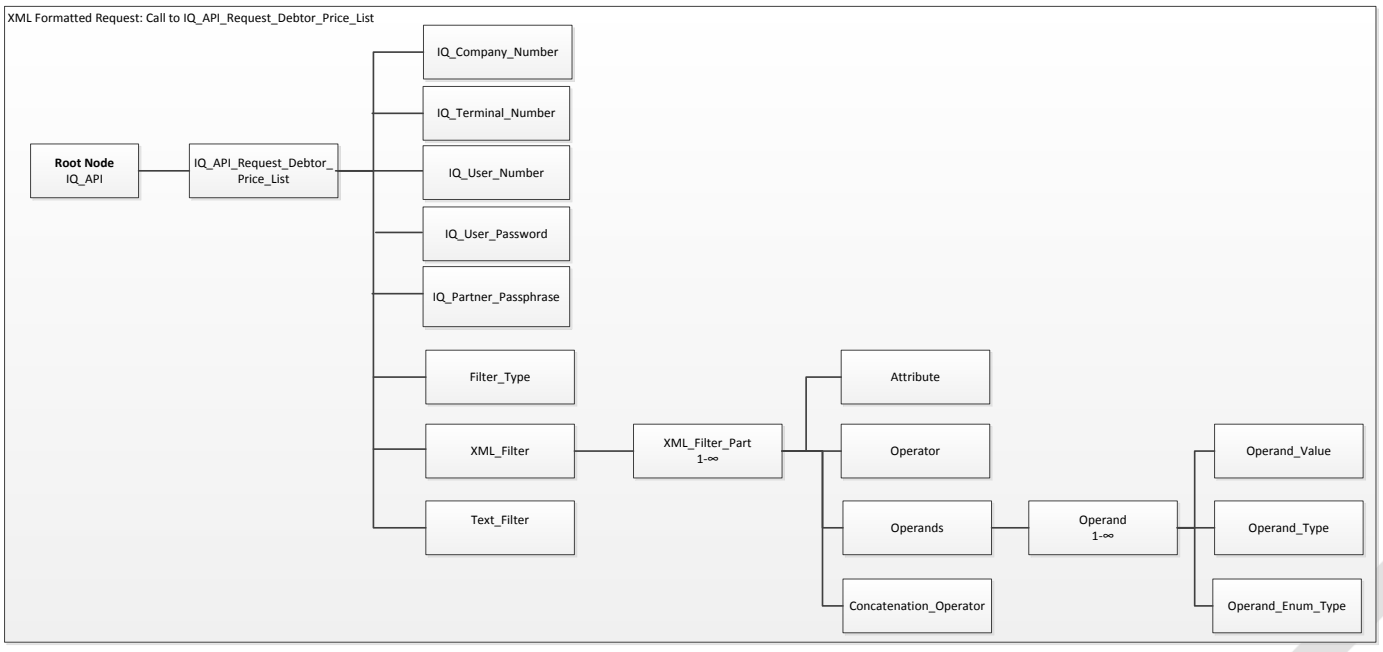
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Debtor_Price_List.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:
Note: This example shows a Debtor Price List Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Debtor_Price_List>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents

can be found in the Appendix section of this guide. For this function call, please see the **TIQStockPriceListMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnDebtorPriceListRequestClick (Sender: TObject);
var
  LProc     : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
  LFreeProc : TIQ_EntAPI_Free_PChar;                    //FreePChar Procedure
  LResult   : TIQ_Type_Char_Result;                     //Result Parameter Value
  LLength   : TIQ_Type_Param_Length;                    //Result Parameter Length
  LSend     : TIQ_Type_Char_Param;                      //Request Parameter
  LXml      : TNativeXml;
  LNode     : TXmlNode;
  LSendL    : Integer;                                  //Length or Request Parameter
  LPassword : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
  try
    LProc     := GetProcAddress(FHandle, 'IQ_API_Request_Debtor_Price_List);
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) Then Exit;

      ....
      //Your code that generates the XML formatted request parameter
      ....
  finally
    ReleaseDLL;
  end;
end;
```

Example Code [C#]:

```
private void btnDebtorPriceListRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Debtor_Price_List (LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Creditor_Price_List

Description: This method allows the client application to request Creditor Price List(s) attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]
TIQ_EntAPI_Request_Procedure = Function(aParam : TIQ_Type_Char_Param;

```
                                    aParam_Length       : TIQ_Type_Param_Length;
                                    out aResult         : TIQ_Type_Char_Result;
                                    var aResult_Length  : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Creditor_Price_List")]
    public static extern int IQ_API_Request_Creditor_Price_List([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr
aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

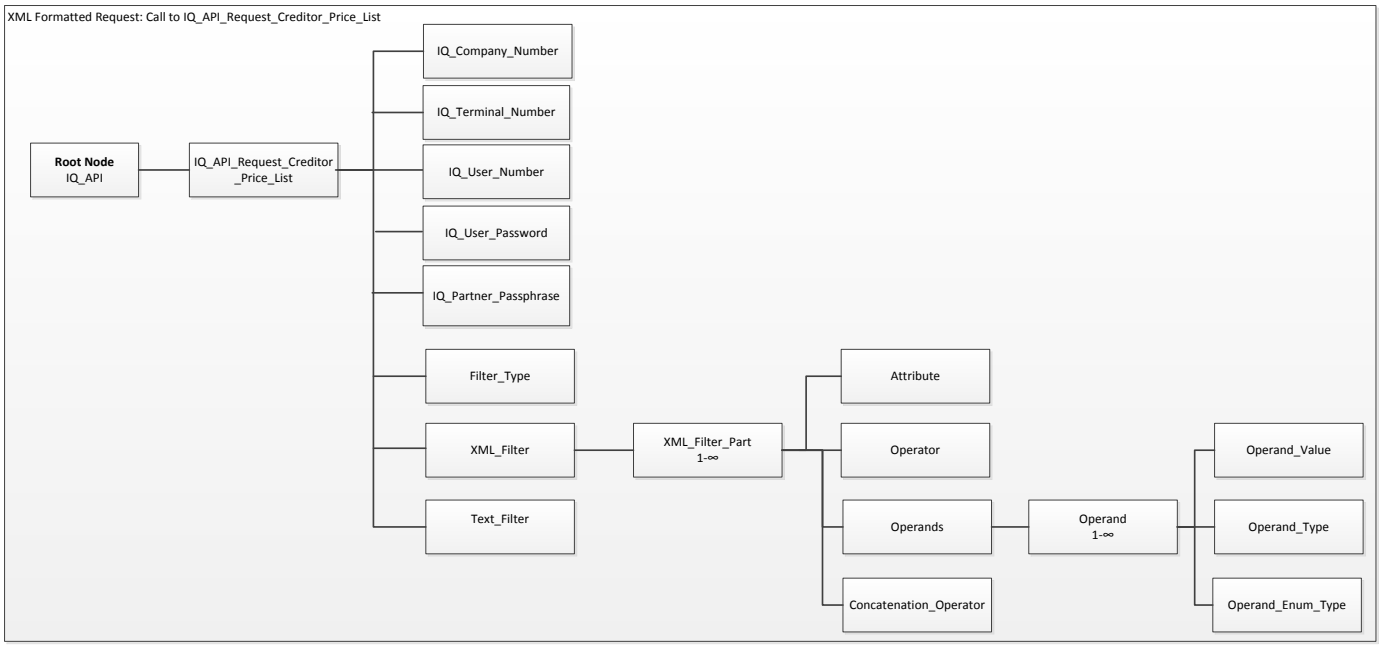
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Creditor_Price_List.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:

Note: This example shows a Creditor Price List Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Creditor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </IQ_API_Request_Creditor_Price_List>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQStockPriceListMaster** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnCreditorPriceListRequestClick (Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;                //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;                 //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;                //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                  //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                              //Length or Request Parameter
 LPassword : String;
begin
```

```
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Creditor_Price_List);
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

   ....
   //Your code that generates the XML formatted request parameter
   ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCreditorPriceListRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

  ....
  //Your code that generates the XML formatted request parameter
  //and stores it in LMessage and its length in FMessageLength
  ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Creditor_Price_List (LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);
```

```
 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Debtor_Store_Departments_Associations

Description: This method allows the client application to request Debtor Store Department Association attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Request_Procedure = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                        out aResult           : TIQ_Type_Char_Result;
                                        var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Debtor_Store_Departments_Associations")]
    public static extern int IQ_API_Request_Debtor_Store_Departments_Associations ([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of

the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

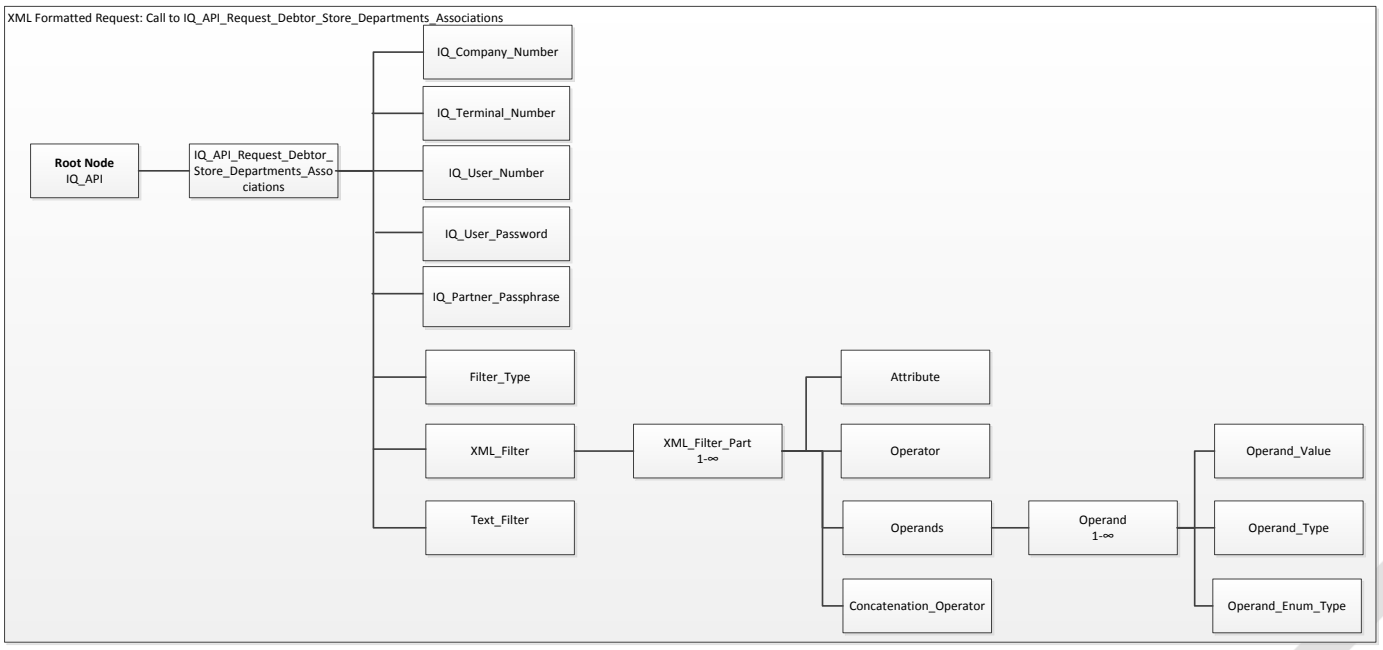
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Debtor_Store_Departments_Associations.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:
Note: This example shows a Debtor Store Department Assocation Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 < IQ_API_Request_Debtor_Store_Departments_Associations>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Debtor_Store_Departments_Associations>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the

TIQStoreDepartmentAssociation XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnDebtorStoreDeptAssRequestClick (Sender: TObject);
var
  LProc     : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
  LFreeProc : TIQ_EntAPI_Free_PChar;                    //FreePChar Procedure
  LResult   : TIQ_Type_Char_Result;                     //Result Parameter Value
  LLength   : TIQ_Type_Param_Length;                    //Result Parameter Length
  LSend     : TIQ_Type_Char_Param;                      //Request Parameter
  LXml      : TNativeXml;
  LNode     : TXmlNode;
  LSendL    : Integer;                                  //Length or Request Parameter
  LPassword : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
  try
    LProc     := GetProcAddress(FHandle, 'IQ_API_Request_Debtor_Store_Departments_Associations);
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) Then Exit;

      ....
     //Your code that generates the XML formatted request parameter
      ....
  finally
    ReleaseDLL;
  end;
end;
```

Example Code [C#]:

```
private void btnDebtorStoreDeptAssRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Debtor_Store_Departments_Associations(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);

 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Request_Creditor_Store_Departments_Associations

Description: This method allows the client application to request Creditor Store Department Association attributes and records from the IQ application / related database tables. This method expects 4 (four) parameters of which: The first is the XML formatted request, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]
TIQ_EntAPI_Request_Procedure = Function(aParam : TIQ_Type_Char_Param;

```
                                    aParam_Length       : TIQ_Type_Param_Length;
                                    out aResult         : TIQ_Type_Char_Result;
                                    var aResult_Length  : TIQ_Type_Param_Length): TIQ_Type_Result;
        stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Request_Creditor_Store_Departments_Associations")]
    public static extern int IQ_API_Request_Creditor_Store_Departments_Associations ([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted request.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

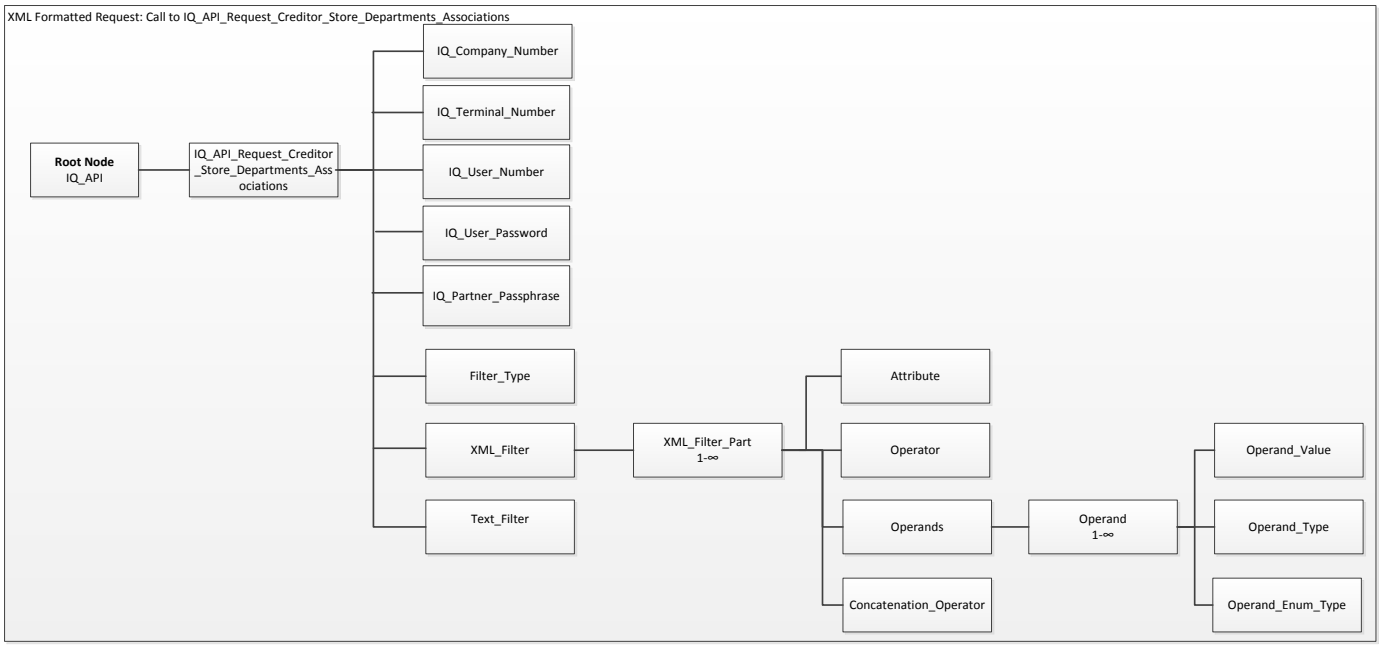
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Request:

The XML formatted request is generated by the host application and contains information specifying the attributes and range of records required by the host application. It also contains the IQ Company from which this information should be extracted. This request currently allows requests to one company at a time.



This request type is identified by the first child node of the XML Root Node and should be named **IQ_API_Request_Creditor_Store_Departments_Associations.**

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

XML Request Example:

Note: This example shows a Creditor Store Department Assocation Request from Company 001.

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 < IQ_API_Request_Creditor_Store_Departments_Associations>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
 </ IQ_API_Request_Creditor_Store_Departments_Associations>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the following two subnodes (at least).

1) IQ_API_Error
2) IQ_API_Result_Data: An XML formatted result containing the Creditor Store Department Records and Attributes as requested per your formatted XML Request parameter. The format of these IQ XML Documents can be found in the Appendix section of this guide. For this function call, please see the **TIQStoreDepartmentAssociation** XML Schema. Note that the XML response can contain ONE or MANY results.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest. btnCreditorStoreDeptAssRequestClick (Sender: TObject);
var
 LProc    : TIQ_EntAPI_Request_Procedure;  //Request Data Procedure
 LFreeProc : TIQ_EntAPI_Free_PChar;              //FreePChar Procedure
 LResult   : TIQ_Type_Char_Result;               //Result Parameter Value
 LLength   : TIQ_Type_Param_Length;              //Result Parameter Length
 LSend     : TIQ_Type_Char_Param;                //Request Parameter
 LXml      : TNativeXml;
 LNode     : TXmlNode;
 LSendL    : Integer;                            //Length or Request Parameter
 LPassword : String;
```

```
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; //Handle to DLL Stored in FHandle, declared outside this method
 try
  LProc    := GetProcAddress(FHandle, 'IQ_API_Request_Creditor_Store_Departments_Associations);
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) Then Exit;

    ....
    //Your code that generates the XML formatted request parameter
    ....
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCreditorStoreDeptAssRequest_Click(object sender, EventArgs e)
{
 IntPtr LResult;
 string LResultString;
 int LResultLength;
 string LMessage;
 int LMessageLength;
 int LCallResult;

   ....
   //Your code that generates the XML formatted request parameter
   //and stores it in LMessage and its length in FMessageLength
   ....
 LResultLength  = 0;

 LCallResult = IQ_API_Request_Creditor_Store_Departments_Associations(LMessage, LMessageLength, out LResult, ref LResultLength);
 LResultString = Marshal.PtrToStringAnsi(LResult);
```

```
 if (LCallResult != 0)
 {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
 }

 lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Submit_Stock_Attributes

Description: This method allows the host application to submit Stock Item Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Stock_Attributes")]public static extern int IQ_API_Submit_Stock_Attributes([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

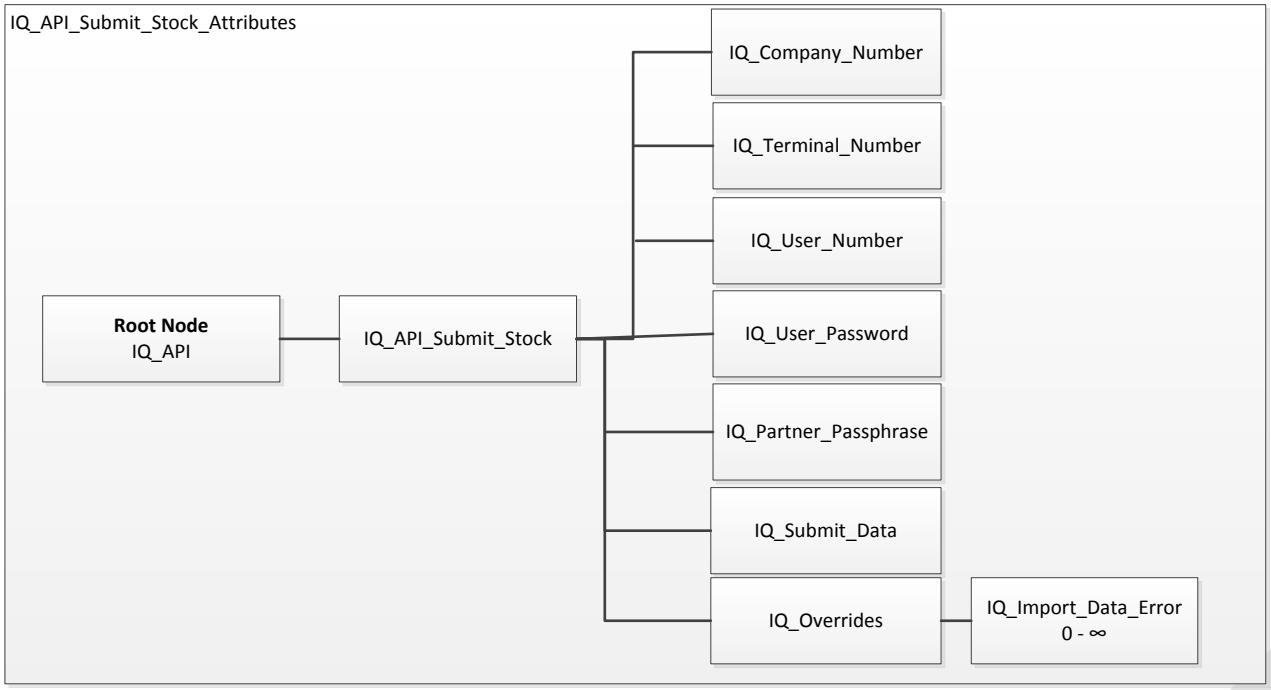
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Stock node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from to which the provided data will be submitted.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Stock related submissions. It contains the root node, the identifying node <IQ_API_Submit_Stock>, the relevant Company Number, the actual submission data and a single override for imeDuplicateCode which will allow importing of Stock even if the Stock Code already exists (such existing stock code will then be updated instead of attempting to create a new stock code). This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Stock>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides>
  <IQ_Import_Data_Error>imeDuplicateCode</IQ_Import_Data_Error>
  <IQ_Overrides>
 </IQ_API_Submit_Stock>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnStockSubmitClick(Sender: TObject);
var
 FProc: TIQ_EntAPI_Submit_Procedure;
 FFreeProc: TIQ_EntAPI_Free_PChar;
 FXMLString: PChar;
 FXMLSource: TNativeXML;
 FXML: TNativeXML;
 FXMLLength: Integer;
 FRes: PChar;
 FResLength: Integer;
 FNewRoot: TXMLNode;
 FOverrideNode: TXMLNode;
 FFileName: String;
 FSubmitDataNode: TXMLNode;
begin
```

```
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  FProc := GetProcAddress(FHandle,'IQ_API_Submit_Stock_Attributes');
  FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

  If Not Assigned(FProc) Then Exit;
  If not Assigned(FFreeProc) Then Exit;

  //..
        //Your code to generate the submission data according to XML Schema
  //..

  FProc(FXMLString, FXMLLength, FRes, FResLength);

  SetResult(Copy(Fres, 1, Freslength)); //Shows result in Memo Component
  FFreeProc(FRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnSubmitStock_Click(object sender, EventArgs e)
 {

        //..
            //Your code to generate the submission data according to XML Schema
        //..

   string FMessage;
   int FMessageLength;
   IntPtr FResult;
   string FResultString;
   int FResultLength = 0;
   int FCallResult;
```

```
    FMessage = FStringWriter.ToString(); //XML Submission Data
    FMessageLength = FMessage.Length;    //Length of it

    FCallResult = IQ_API_Submit_Stock_Attributes(FMessage, FMessageLength, out FResult, ref
                                  FResultLength);

    FResultString = Marshal.PtrToStringAnsi(FResult);

    if (FCallResult != 0)
    {
       MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
    }
     lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
  }
```

IQ_API_Submit_Debtor_Attributes

Description: This method allows the host application to submit Debtor Item Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                  : TIQ_Type_Char_Param;
                                        aParam_Length          : TIQ_Type_Param_Length;
                                       Out aResult             : TIQ_Type_Char_Result;
                                        Var aResult_Length     : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Debtor_Attributes")]public static extern int IQ_API_Submit_Debtor_Attributes([MarshalAs(UnmanagedType.LPStr)]string aParam,
int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:
aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:
aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

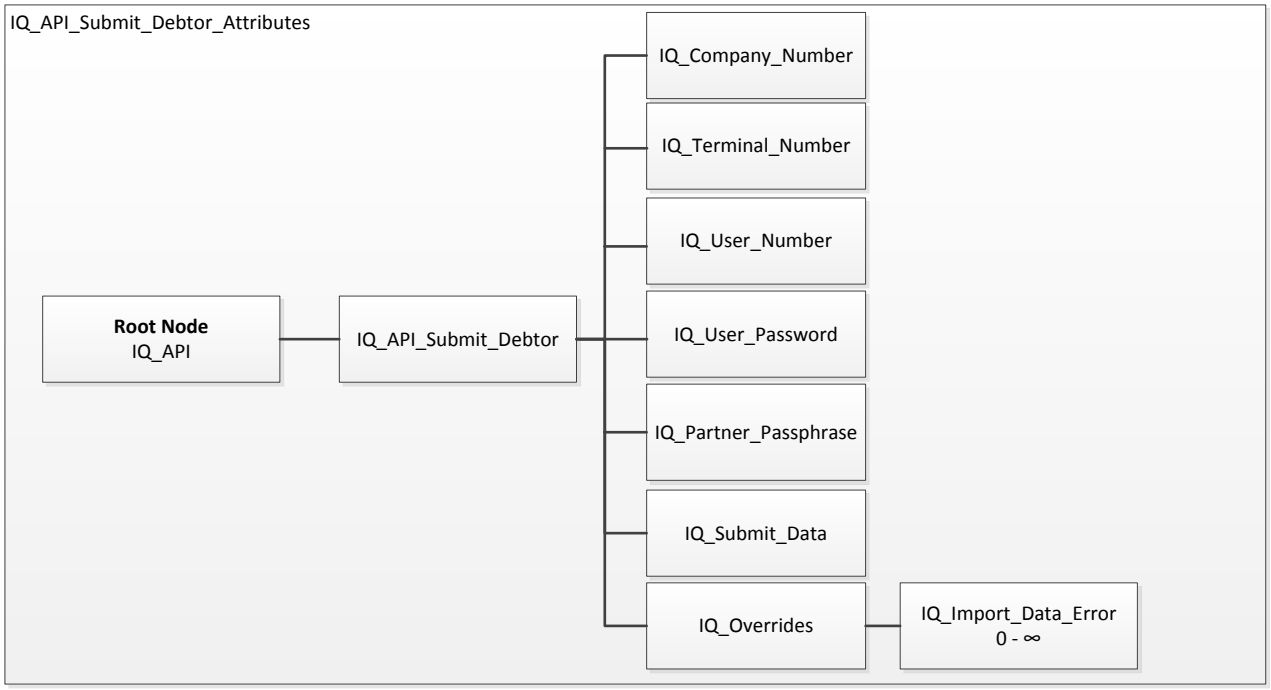
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Debtor node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from to which the provided data will be submitted.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Debtor related submissions. It contains the root node, the identifying node <IQ_API_Submit_Debtor>, the relevant Company Number, the actual submission data and a single override for imeDuplicateAccount which will allow importing of Debtor even if the Debtor's Account already exists (such existing account will then be updated instead of attempting to create a new account). This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>

 <IQ_API_Submit_Debtor>

  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>

  <IQ_User_Number>1</IQ_User_Number>

  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>

  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>

  <IQ_Submit_Data />
  <IQ_Overrides>
  <IQ_Import_Data_Error>imeDuplicateAccount</IQ_Import_Data_Error>
  <IQ_Overrides>

 </IQ_API_Submit_Stock>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDebtorSubmitClick(Sender: TObject);
var
 FProc: TIQ_EntAPI_Submit_Procedure;
 FFreeProc: TIQ_EntAPI_Free_PChar;
 FXMLString: PChar;
 FXMLSource: TNativeXML;
 FXML: TNativeXML;
 FXMLLength: Integer;
 FRes: PChar;
 FResLength: Integer;
 FNewRoot: TXMLNode;
 FOverrideNode: TXMLNode;
 FFileName: String;
 FSubmitDataNode: TXMLNode;
begin
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  FProc := GetProcAddress(FHandle,'IQ_API_Submit_Debtor_Attributes');
  FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

  If Not Assigned(FProc) Then Exit;
  If not Assigned(FFreeProc) Then Exit;

  //..
        //Your code to generate the submission data according to XML Schema
  //..

  FProc(FXMLString, FXMLLength, FRes, FResLength);

  SetResult(Copy(Fres, 1, Freslength)); //Shows result in Memo Component
  FFreeProc(FRes);
 finally
  ReleaseDLL;
 end;
```

```
end;
```

Example Code [C#]:

```
private void btnSubmitDebtor_Click(object sender, EventArgs e)
 {

            //..
                    //Your code to generate the submission data according to XML Schema
            //..

    string FMessage;
    int FMessageLength;
    IntPtr FResult;
    string FResultString;
    int FResultLength = 0;
    int FCallResult;

    FMessage = FStringWriter.ToString(); //XML Submission Data
    FMessageLength = FMessage.Length;    //Length of it

    FCallResult = IQ_API_Submit_Debtor_Attributes(FMessage, FMessageLength, out FResult, ref
                                FResultLength);

    FResultString = Marshal.PtrToStringAnsi(FResult);

    if (FCallResult != 0)
    {
      MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
    }
     lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
 }
```

IQ_API_Submit_Document_Sales_Order

Description: This method allows the host application to submit Sales Order Documents to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the

length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                    : TIQ_Type_Char_Param;
                                        aParam_Length            : TIQ_Type_Param_Length;
                                       Out aResult               : TIQ_Type_Char_Result;
                                        Var aResult_Length       : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Document_Sales_Order")]public static extern int
IQ_API_Submit_Document_Sales_Order([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int
aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

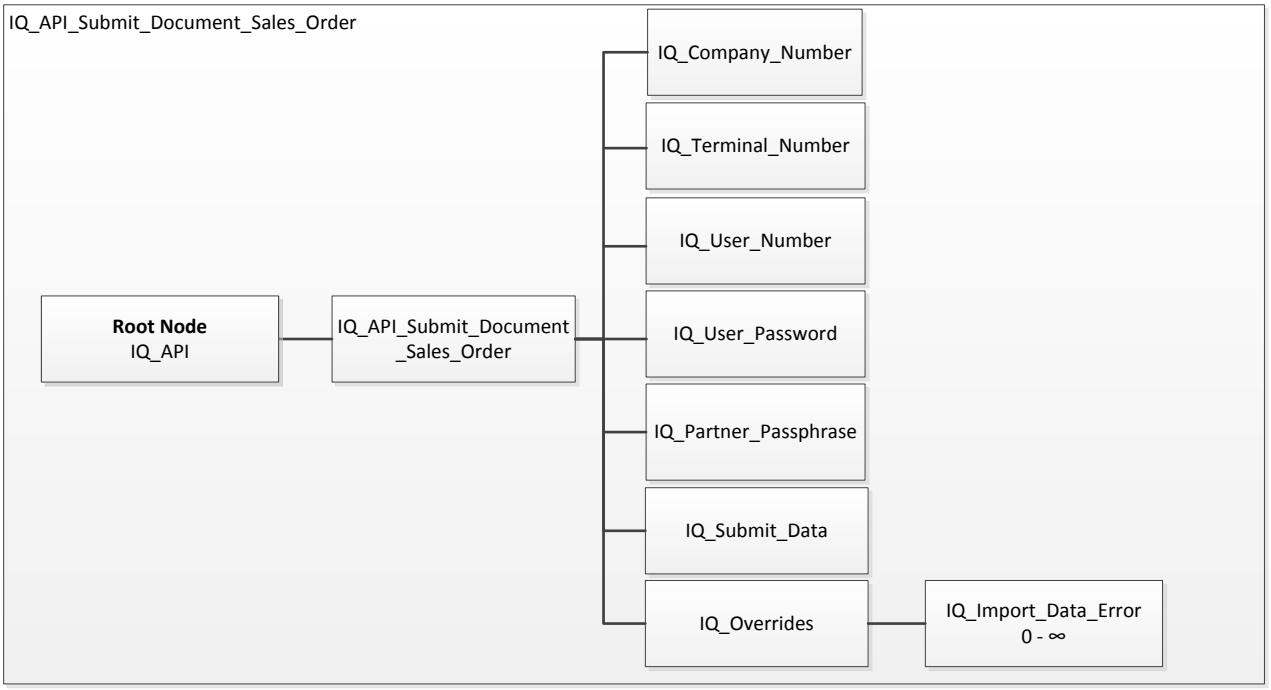
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Document_Sales_Order node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from to which the provided data will be submitted.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQDocumentError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Sales Order related submissions. It contains the root node, the identifying node <IQ_API_Submit_Document_Sales_Order>, the relevant Company Number, the actual submission data and a single override for ideNegativeStock which will allow importing of Sales Order documents only if the Negative Stock restriction is approved / overridden by the host application. If not, the IQ_API_Error will return this as Extended Data in one of the IQ_API_Error items.

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Document_Sales_Order>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Overrides>
   <IQ_Import_Data_Error>ideNegativeStock</IQ_Import_Data_Error>
  </IQ_Overrides>
  <IQ_Submit_Data />
 </IQ_API_Submit_Document_Sales_Order>
```

</IQ_API>

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnSORSubmitClick(Sender: TObject);
var
  FProc: TIQ_EntAPI_Submit_Procedure;
  FFreeProc: TIQ_EntAPI_Free_PChar;
  FXMLString: PChar;
  FXMLSource: TNativeXML;
  FXML: TNativeXML;
  FXMLLength: Integer;
  FRes: PChar;
  FResLength: Integer;
  FNewRoot: TXMLNode;
  FOverrideNode: TXMLNode;
  FFileName: String;
  FSubmitDataNode: TXMLNode;
begin
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    FProc := GetProcAddress(FHandle,'IQ_API_Submit_Document_Sales_Order');
    FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

    If Not Assigned(FProc) Then Exit;
    If not Assigned(FFreeProc) Then Exit;

    //..
            //Your code to generate the submission data according to XML Schema
    //..

    FProc(FXMLString, FXMLLength, FRes, FResLength);
```

```
  SetResult(Copy(Fres, 1, Freslength)); //Shows result in Memo Component
  FFreeProc(FRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnSubmitSOR_Click(object sender, EventArgs e)
 {

        //..
            //Your code to generate the submission data according to XML Schema
        //..

   string FMessage;
   int FMessageLength;
   IntPtr FResult;
   string FResultString;
   int FResultLength = 0;
   int FCallResult;

   FMessage = FStringWriter.ToString(); //XML Submission Data
   FMessageLength = FMessage.Length;    //Length of it

   FCallResult = IQ_API_Submit_Document_Sales_Order(FMessage, FMessageLength, out FResult, ref
                                FResultLength);

   FResultString = Marshal.PtrToStringAnsi(FResult);

   if (FCallResult != 0)
   {
     MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
   }
   lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
```

}

IQ_API_Submit_Document_Purchase_Order

Description: This method allows the host application to submit Purchase Order Documents to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                    : TIQ_Type_Char_Param;
                                        aParam_Length            : TIQ_Type_Param_Length;
                                        Out aResult              : TIQ_Type_Char_Result;
                                        Var aResult_Length       : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Document_Purchase_Order")]public static extern int
IQ_API_Submit_Document_Purchase_Order([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int
aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

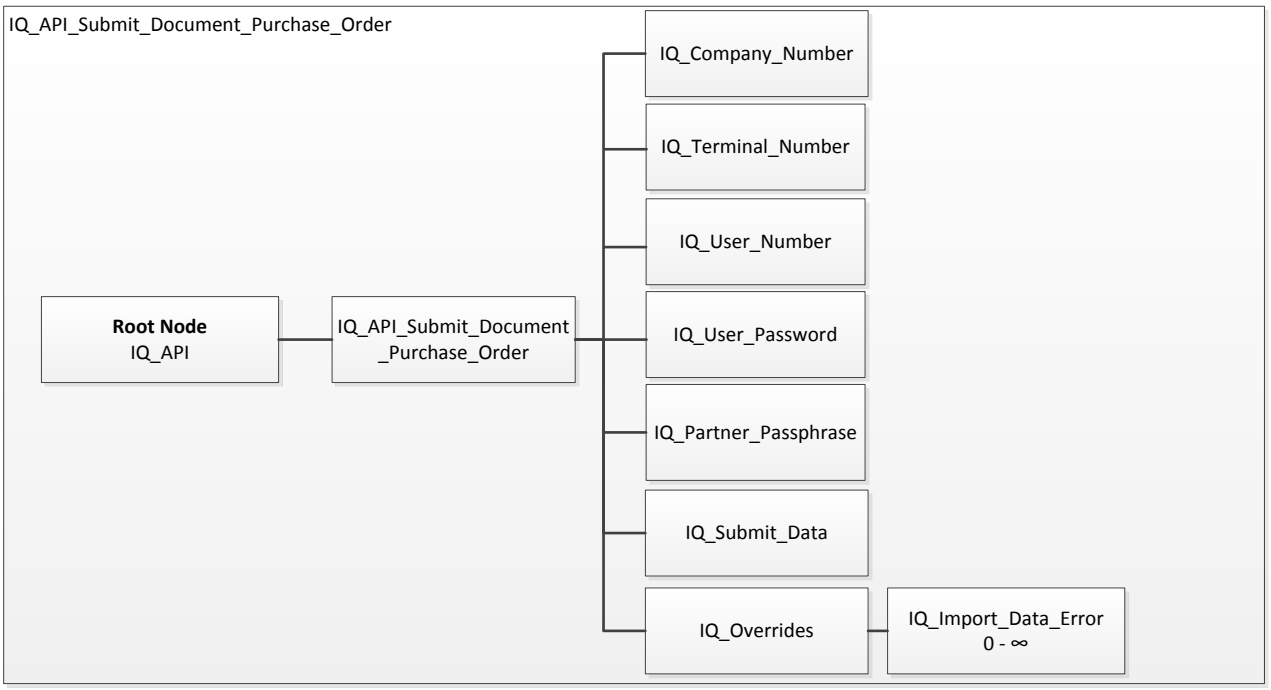
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Document_Purchase_Order node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from to which the provided data will be submitted.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQDocumentError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Purchase Order related submissions. It contains the root node, the identifying node <IQ_API_Submit_Document_Purchase_Order>, the relevant Company Number, the actual submission data and a single override for ideNegativeStock which will allow importing of Purchase Order documents only if the Negative Stock restriction is approved / overridden by the host application. If not, the IQ_API_Error will return this as Extended Data in one of the IQ_API_Error items.

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Document_Purchase_Order>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
```

```
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Overrides>
   <IQ_Import_Data_Error>ideNegativeStock</IQ_Import_Data_Error>
  </IQ_Overrides>
  <IQ_Submit_Data />
 </IQ_API_Submit_Document_Sales_Order>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnPORSubmitClick(Sender: TObject);
var
 FProc: TIQ_EntAPI_Submit_Procedure;
 FFreeProc: TIQ_EntAPI_Free_PChar;
 FXMLString: PChar;
 FXMLSource: TNativeXML;
 FXML: TNativeXML;
 FXMLLength: Integer;
 FRes: PChar;
 FResLength: Integer;
 FNewRoot: TXMLNode;
 FOverrideNode: TXMLNode;
 FFileName: String;
 FSubmitDataNode: TXMLNode;
begin
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  FProc := GetProcAddress(FHandle,'IQ_API_Submit_Document_Purchase_Order');
  FFreeProc := GetprocAddress(FHandle, 'IQ_API_Free_PChar');

  If Not Assigned(FProc) Then Exit;
  If not Assigned(FFreeProc) Then Exit;
```

```
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  FProc(FXMLString, FXMLLength, FRes, FResLength);

  SetResult(Copy(Fres, 1, Freslength)); //Shows result in Memo Component
  FFreeProc(FRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnSubmitPOR_Click(object sender, EventArgs e)
 {

        //..
            //Your code to generate the submission data according to XML Schema
        //..

   string FMessage;
   int FMessageLength;
   IntPtr FResult;
   string FResultString;
   int FResultLength = 0;
   int FCallResult;

   FMessage = FStringWriter.ToString(); //XML Submission Data
   FMessageLength = FMessage.Length;    //Length of it

   FCallResult = IQ_API_Submit_Document_Purchase_Order(FMessage, FMessageLength, out FResult, ref
                        FResultLength);

   FResultString = Marshal.PtrToStringAnsi(FResult);
```

```
    if (FCallResult != 0)
    {
      MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
    }
    lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
  }
```

IQ_API_Submit_Document_Invoice

Description: This method allows the host application to submit Invoice Documents to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                  : TIQ_Type_Char_Param;
                                        aParam_Length          : TIQ_Type_Param_Length;
                                        Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length     : TIQ_Type_Param_Length): TIQ_Type_Result;
                    stdcall;
```

[C#]

```
[DllImport(@"..\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Document_Invoice")]public static extern int IQ_API_Submit_Document_Invoice([MarshalAs(UnmanagedType.LPStr)]string aParam,
int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

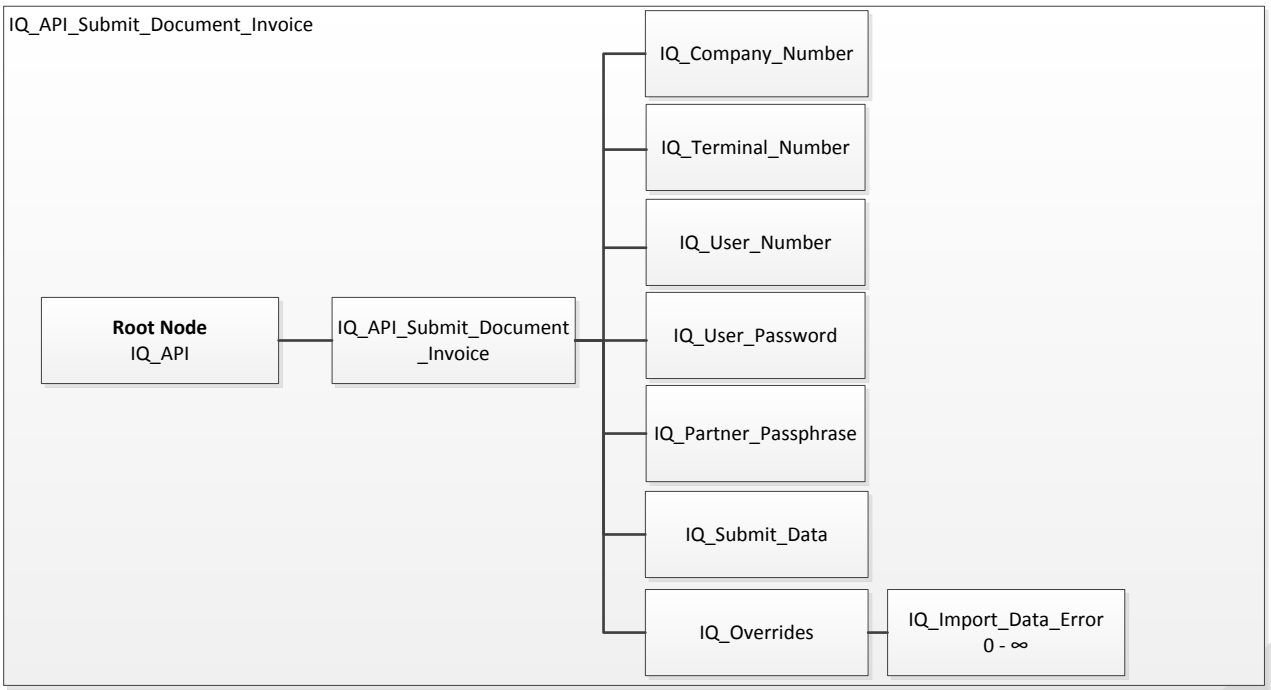
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Document_Invoice** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQDocumentError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Invoice related submissions. It contains the root node, the identifying node <IQ_API_Submit_Document_Invoice>, the relevant Company Number, the actual submission data and a single override for ideNegativeStock which will allow importing of Invoice documents only if the Negative Stock restriction is approved / overridden by the host application. If not, the IQ_API_Error will return this as Extended Data in one of the IQ_API_Error items.

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
```

```
<IQ_API_Submit_Document_Invoice>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Overrides>
    <IQ_Import_Data_Error>ideNegativeStock</IQ_Import_Data_Error>
  </IQ_Overrides>
  <IQ_Submit_Data />
 </IQ_API_Submit_Document_Invoice>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.Button2Click(Sender: TObject);
var
  FProc: TIQ_EntAPI_Request_Procedure;
  FFreeProc: TIQ_EntAPI_Free_PChar;
  FXMLString: PChar;
  FXMLLength: Integer;
  FRes: PChar;
  FResLength: Integer;
  FXML: TNativeXML;
  FNewRoot: TXMLNode;
  FFileName: String;
  FXMLSource : TNativeXML;
  FSubmitDataNode: TXMLNode;
  FOverrideNode: TXMLNode;
  FResInt: Integer;
begin
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    FProc := GetProcAddress(FHandle,'IQ_API_Submit_Document_Invoice');
```

```
FFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

If Not Assigned(FProc) Then Exit;
If not Assigned(FFreeProc) Then Exit;

FFileName := '';
ShowMessage('Please select a valid Invoice IQXML File');
if not dlgOpen.Execute Then Exit;
FFileName := dlgOpen.Filename;
if Length(FFilename) = 0 Then Exit;

FXML := TNativeXML.Create;
FXMLSource := TNativeXML.Create;
try
 FXMLSource.LoadFromFile(FFileName);

 FXML.Root.Name := 'IQ_API';
 FNewRoot := FXML.Root.NodeNew('IQ_API_Submit_Document_Invoice');
 FNewRoot.NodeNew('IQ_Company_Number').ValueAsString := '001';
 FSubmitDataNode := FNewRoot.NodeNew('IQ_Submit_Data').NodeNew('IQ_Root_XML');
 FSubmitDataNode.Assign(FXMLSource.Root);

 FOverrideNode := FNewRoot.NodeNew('IQ_Overrides');
 FOverrideNode.NodeNew('IQ_Import_Data_Error').ValueAsString := 'ideNegativeStock';
 FXMLString := PChar(FXML.WriteToString);
 FXMLLength := Length(FXMLString);
finally
 FreeAndNil(FXML);
 FreeAndNil(FXMLSource);
end;

FResInt := FProc(FXMLString, FXMLLength, FRes, FResLength);
ShowMessage(IntToStr(FResInt));
SetResult(Copy(Fres, 1, Freslength));
FFreeProc(FRes);
finally
 ReleaseDLL;
end;
```

```
end;
```

Example Code [C#]:

```
private void btnSubmitInv_Click(object sender, EventArgs e)
{
  string FFileName = GetFileName("Please select a valid Invoice IQXML File", "File Selection");

  if (!File.Exists(FFileName))
  {
    MessageBox.Show("File selected does not exist.");
    return;
  }

  XmlReader FReader = XmlReader.Create(FFileName);
  StringWriter FStringWriter = new StringWriter();

  using (XmlWriter FWriter = XmlWriter.Create(FStringWriter))
  {
    //Create IQ_API XML String
    FWriter.WriteStartDocument();
    FWriter.WriteStartElement("IQ_API");
    FWriter.WriteStartElement("IQ_API_Submit_Document_Invoice");
    FWriter.WriteElementString("IQ_Company_Number", "001");
    FWriter.WriteStartElement("IQ_Overrides");
    FWriter.WriteElementString("IQ_Import_Data_Error", "ideNegativeStock");
    FWriter.WriteEndElement();
    FWriter.WriteStartElement("IQ_Submit_Data");

    //Add Nodes From Document Loaded
    CopyXML(FReader, FWriter);

    FWriter.WriteEndElement();
    FWriter.WriteEndElement();
    FWriter.WriteEndElement();
```

```
    FWriter.WriteEndDocument();

    FWriter.Flush();
    //Using does Close Automatically for FWriter
}

FReader.Close();
//lstResult.Text = FStringWriter.ToString();
//return;

string FMessage;
int FMessageLength;
IntPtr FResult;
string FResultString;
int FResultLength = 0;
int FCallResult;

FMessage = FStringWriter.ToString();
FMessageLength = FMessage.Length;

FCallResult = IQ_API_Submit_Document_Invoice(FMessage, FMessageLength, out FResult, ref FResultLength);
FResultString = Marshal.PtrToStringAnsi(FResult);

if (FCallResult != 0)
{
    MessageBox.Show("An Error Occurred. Error Code [" + FCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(FResultString.Substring(0, FResultLength));
                                                                    }
```

IQ_API_Submit_Debtor_Journal

Description: This method allows the host application to submit Debtor Journal Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Submit_Debtor_Journal")] public static extern int IQ_API_Submit_Debtor_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

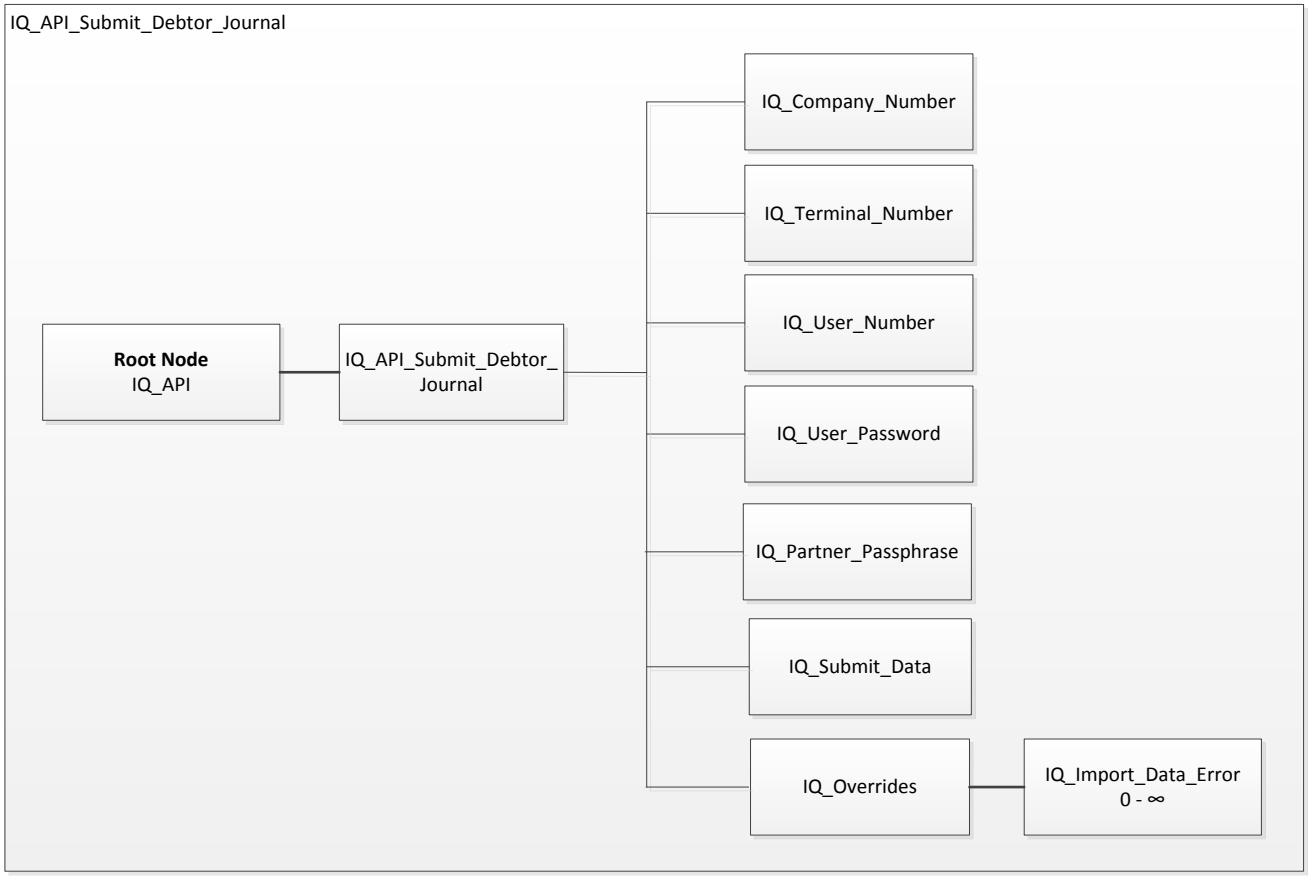
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Debtor_Journal** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQJournalError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Debtor Journal related submissions. It contains the root node, the identifying node <IQ_API_Submit_Debtor_Journal>, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Debtor_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
```

```
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Debtor_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDJournalSubmitClick(Sender: TObject);
var
 LProc        : TIQ_EntAPI_Submit_Procedure;
 LFreeProc    : TIQ_EntAPI_Free_PChar;
 LFileName    : String;
 LXML         : TNativeXML;
 LXMLSource   : TNativeXML;
 LNewRoot     : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString   : PChar;
 LRes         : PChar;
 LXMLLength   : Integer;
 LResLength   : Integer;
 LResInt      : Integer;
 LPassword    : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Debtor_Journal');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;
```

```
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnDJournalSubmit_Click(object sender, EventArgs e)
{
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Debtor_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
```

```
  }

  lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Submit_Creditor_Journal

Description: This method allows the host application to submit Creditor Journal Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Creditor_Journal")] public static extern int IQ_API_Submit_Creditor_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of

the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

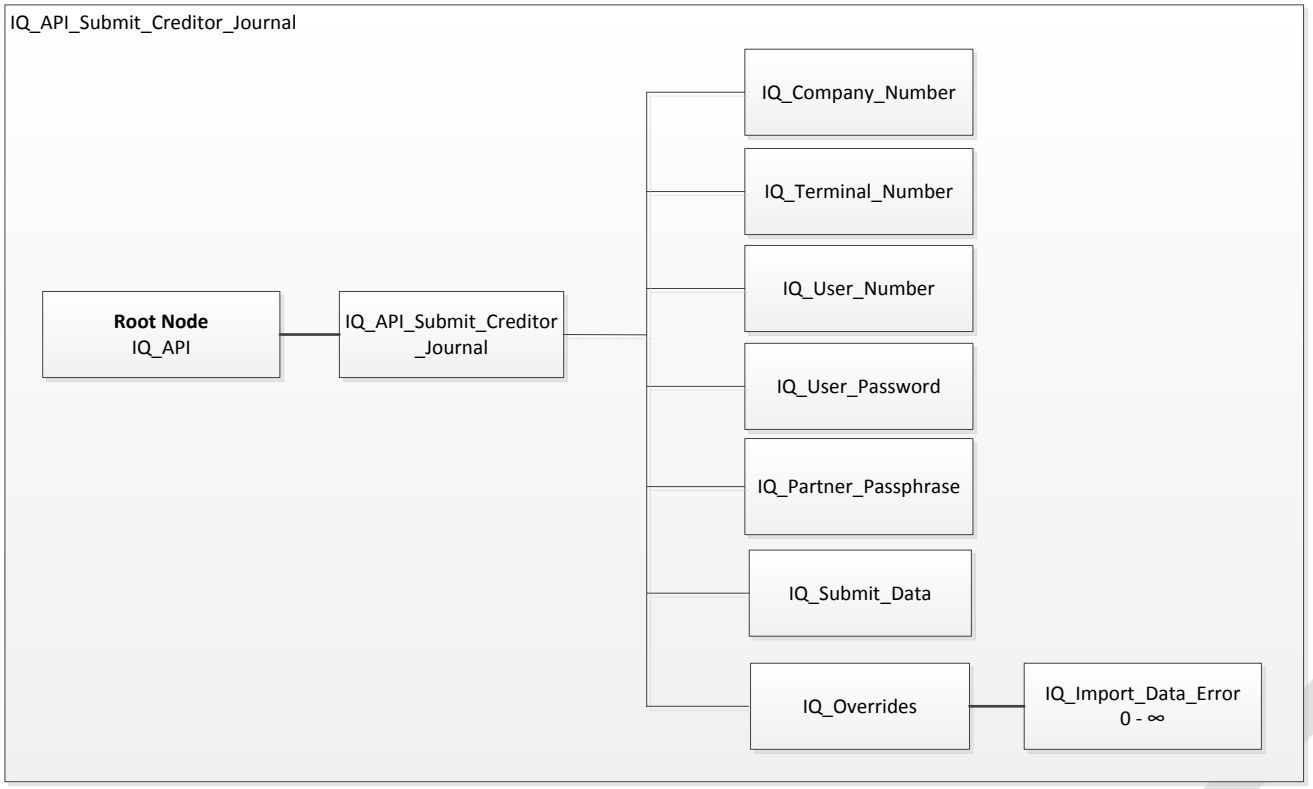
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Creditor_Journal** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQJournalError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Creditor Journal related submissions. It contains the root node, the identifying node <IQ_API_Submit_Creditor_Journal>, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Creditor_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Creditor_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnCJournalSubmitClick(Sender: TObject);
var
 LProc        : TIQ_EntAPI_Submit_Procedure;
 LFreeProc    : TIQ_EntAPI_Free_PChar;
 LFileName    : String;
 LXML         : TNativeXML;
 LXMLSource   : TNativeXML;
 LNewRoot     : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString   : PChar;
 LRes         : PChar;
 LXMLLength   : Integer;
 LResLength   : Integer;
 LResInt      : Integer;
 LPassword    : String;
begin
 LPassword := 'Test';
```

```
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  LProc     := GetProcAddress(FHandle,'IQ_API_Submit_Creditor_Journal');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;

  //..
            //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCJournalSubmit_Click(object sender, EventArgs e)
{
  //..
            //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;
```

```
LCallResult = IQ_API_Submit_Creditor_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Ledger_Journal

Description: This method allows the host application to submit Ledger Journal Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length        : TIQ_Type_Param_Length;
                                       Out aResult           : TIQ_Type_Char_Result;
                                        Var aResult_Length   : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Submit_Ledger_Journal")] public static extern int IQ_API_Submit_Ledger_Journal([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

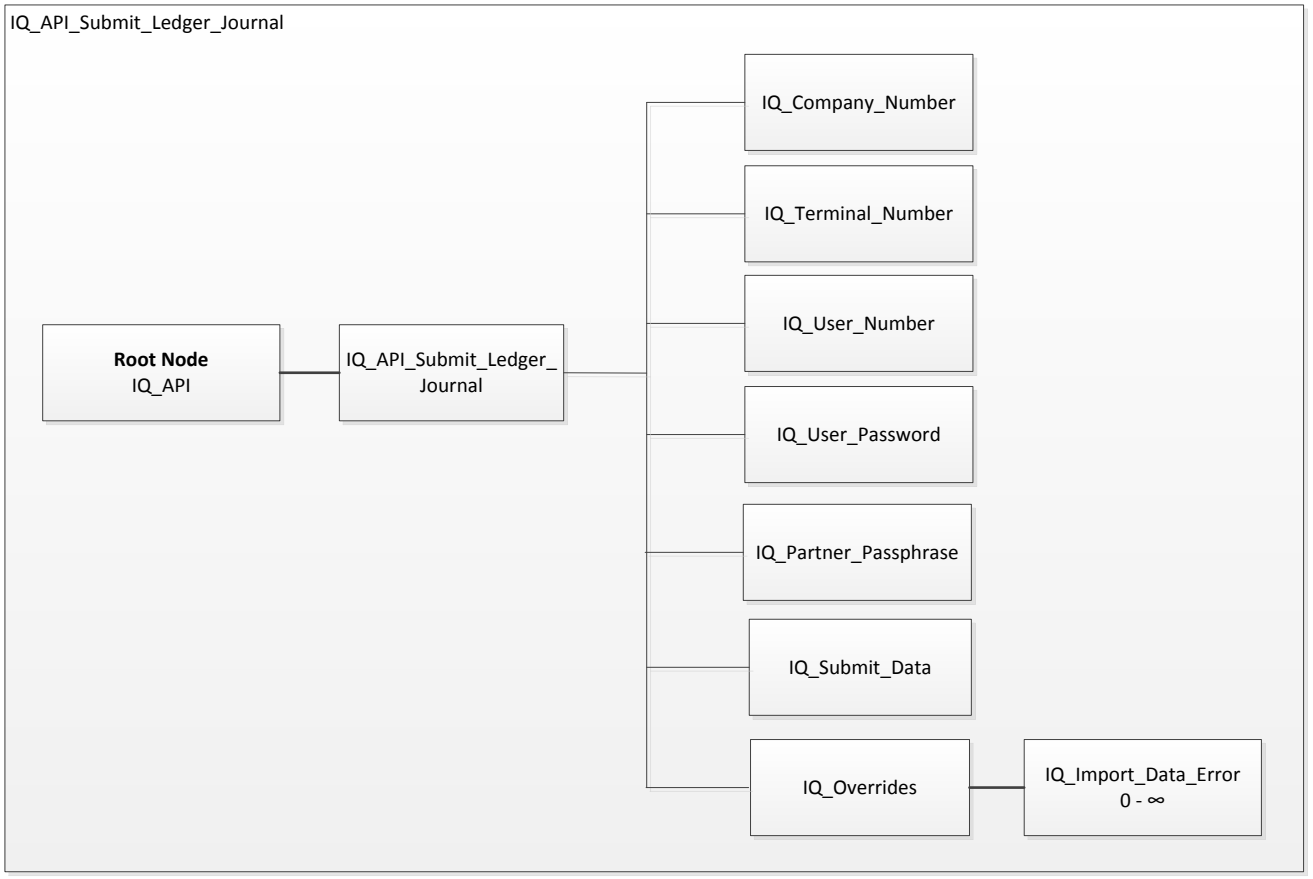
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Ledger_Journal** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQJournalError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Ledger Journal related submissions. It contains the root node, the identifying node <IQ_API_Submit_Ledger_Journal>, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Ledger_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Ledger_Journal>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnLJournalSubmitClick(Sender: TObject);
var
  LProc         : TIQ_EntAPI_Submit_Procedure;
  LFreeProc     : TIQ_EntAPI_Free_PChar;
  LFileName     : String;
  LXML          : TNativeXML;
  LXMLSource    : TNativeXML;
  LNewRoot      : TXMLNode;
  LSubmitDataNode : TXMLNode;
  LXMLString    : PChar;
  LRes          : PChar;
  LXMLLength    : Integer;
  LResLength    : Integer;
  LResInt       : Integer;
  LPassword     : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Ledger_Journal');
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) then Exit;

    //..
            //Your code to generate the submission data according to XML Schema
    //..

    LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
    SetResult(Copy(LRes, 1, LResLength));
    LFreeProc(LRes);
```

```
 finally
   ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnLJournalSubmit_Click(object sender, EventArgs e)
{
  //..
            //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Ledger_Journal(LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Debtor_Store_Departments

Description: This method allows the host application to submit Debtor Store Department Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Debtor_Store_Departments")] public static extern int IQ_API_Submit_Debtor_Store_Departments
([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

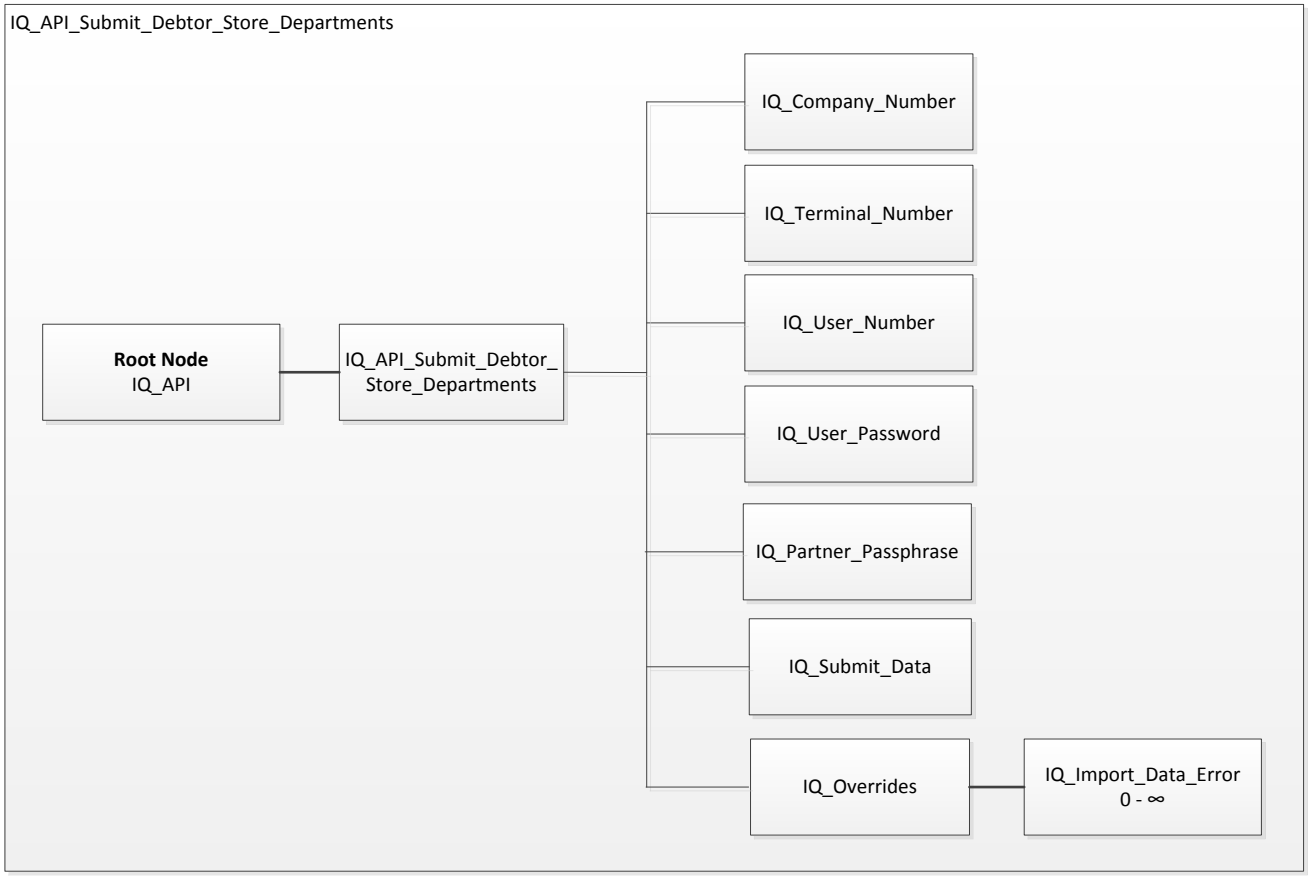
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Debtor_Store_Department** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Debtor Store Department related submissions. It contains the root node, the identifying node **<IQ_API_Submit_Debtor_Store_Department>**, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Debtor_Store_Department>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
```

```
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Debtor_Store_Department >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDStoreDeptSubmitClick(Sender: TObject);
var
 LProc        : TIQ_EntAPI_Submit_Procedure;
 LFreeProc    : TIQ_EntAPI_Free_PChar;
 LFileName    : String;
 LXML         : TNativeXML;
 LXMLSource   : TNativeXML;
 LNewRoot     : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString   : PChar;
 LRes         : PChar;
 LXMLLength   : Integer;
 LResLength   : Integer;
 LResInt      : Integer;
 LPassword    : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Debtor_Store_Departments');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;
```

```
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnDStoreDeptSubmit_Click(object sender, EventArgs e)
{
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Debtor_Store_Departments(LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }
```

```
lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Creditor_Store_Departments

Description: This method allows the host application to submit Creditor Store Department Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Creditor_Store_Departments")] public static extern int IQ_API_Submit_Creditor_Store_Departments
([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

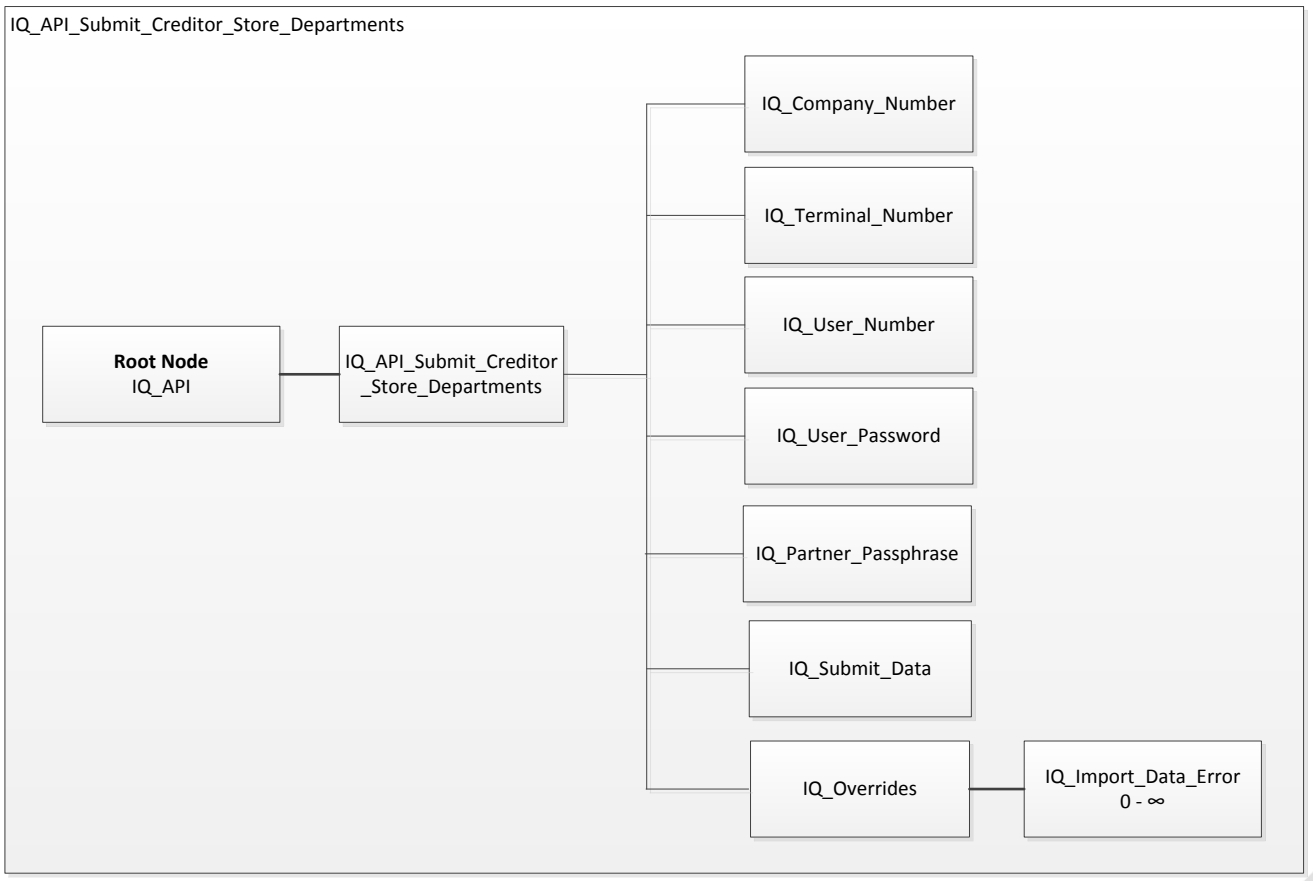
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Creditor_Store_Department** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Creditor Store Department related submissions. It contains the root node, the identifying node **<IQ_API_Submit_Creditor_Store_Department>**, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Creditor_Store_Department>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Creditor_Store_Department >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnCStoreDeptSubmitClick(Sender: TObject);
var
 LProc        : TIQ_EntAPI_Submit_Procedure;
 LFreeProc    : TIQ_EntAPI_Free_PChar;
 LFileName    : String;
 LXML         : TNativeXML;
 LXMLSource   : TNativeXML;
 LNewRoot     : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString   : PChar;
 LRes         : PChar;
 LXMLLength   : Integer;
 LResLength   : Integer;
 LResInt      : Integer;
 LPassword    : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
```

```
 try
   LProc     := GetProcAddress(FHandle,'IQ_API_Submit_Creditor_Store_Departments');
   LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

   if not Assigned(LProc) then Exit;
   if not Assigned(LFreeProc) then Exit;

   //..
            //Your code to generate the submission data according to XML Schema
   //..

   LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
   SetResult(Copy(LRes, 1, LResLength));
   LFreeProc(LRes);
 finally
   ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnCStoreDeptSubmit_Click(object sender, EventArgs e)
{
  //..
            //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;
```

```
LCallResult = IQ_API_Submit_Creditor_Store_Departments(LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Stock_ContractPricing

Description: This method allows the host application to submit Stock Contract Pricing Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length        : TIQ_Type_Param_Length;
                                       Out aResult           : TIQ_Type_Char_Result;
                                        Var aResult_Length   : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Stock_ContractPricing")] public static extern int IQ_API_Submit_Stock_ContractPricing ([MarshalAs(UnmanagedType.LPStr)]string
aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

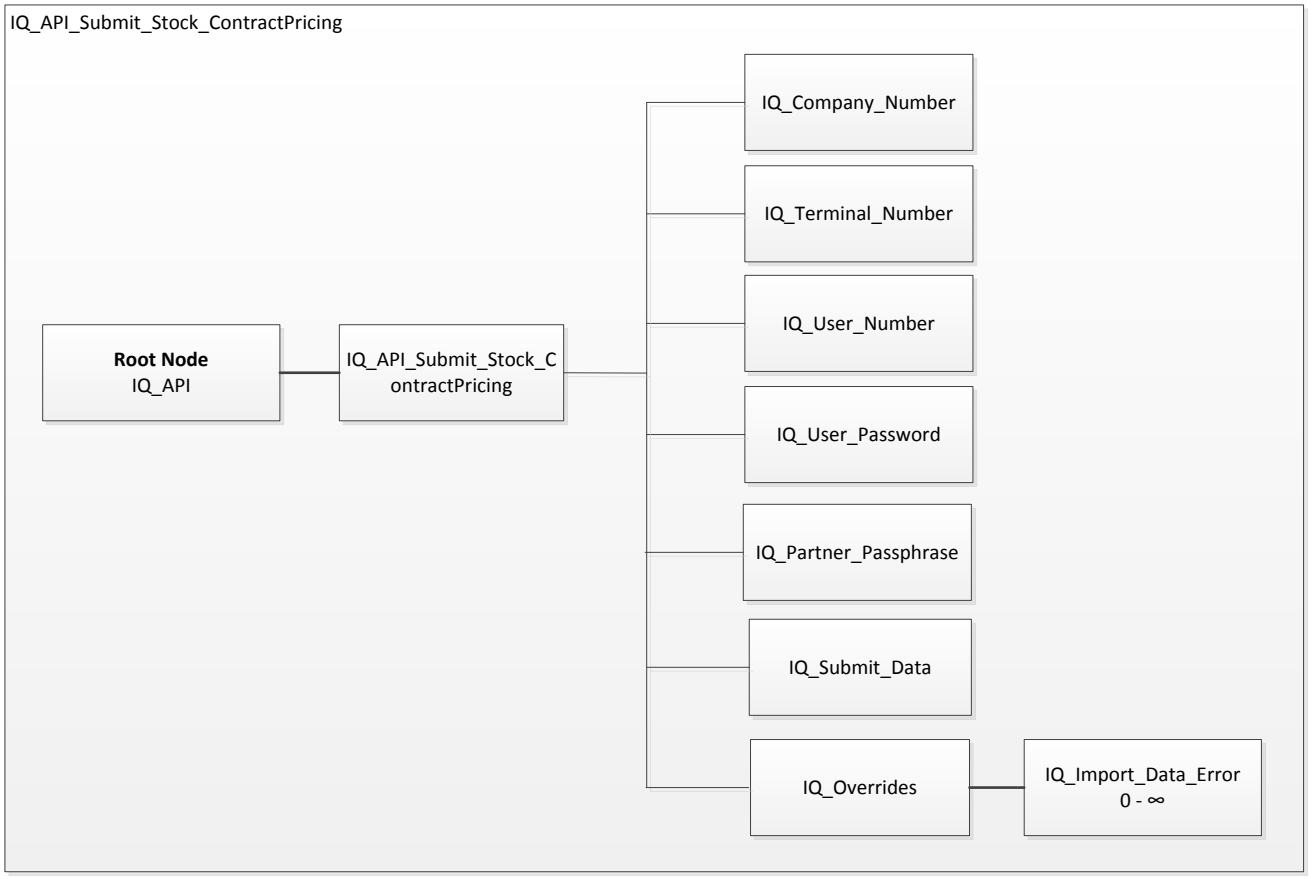
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Stock_ContractPricing** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Stock Contract Pricing related submissions. It contains the root node, the identifying node **< IQ_API_Submit_Stock_ContractPricing>**, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 < IQ_API_Submit_Stock_ContractPricing>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </ IQ_API_Submit_Stock_ContractPricing>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnStockContractSubmitClick(Sender: TObject);
var
  LProc         : TIQ_EntAPI_Submit_Procedure;
  LFreeProc     : TIQ_EntAPI_Free_PChar;
  LFileName     : String;
  LXML          : TNativeXML;
  LXMLSource    : TNativeXML;
  LNewRoot      : TXMLNode;
  LSubmitDataNode : TXMLNode;
  LXMLString    : PChar;
  LRes          : PChar;
  LXMLLength    : Integer;
  LResLength    : Integer;
  LResInt       : Integer;
  LPassword     : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Stock_ContractPricing');
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) then Exit;

    //..
            //Your code to generate the submission data according to XML Schema
    //..

    LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
    SetResult(Copy(LRes, 1, LResLength));
    LFreeProc(LRes);
```

```
 finally
   ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnStockContractSubmit_Click(object sender, EventArgs e)
{
  //..
            //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Stock_ContractPricing (LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Promotion

Description: This method allows the host application to submit Stock Promotion Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                  : TIQ_Type_Char_Param;
                                        aParam_Length          : TIQ_Type_Param_Length;
                                       Out aResult             : TIQ_Type_Char_Result;
                                        Var aResult_Length     : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "
IQ_API_Submit_Promotion")] public static extern int IQ_API_Submit_Promotion ([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

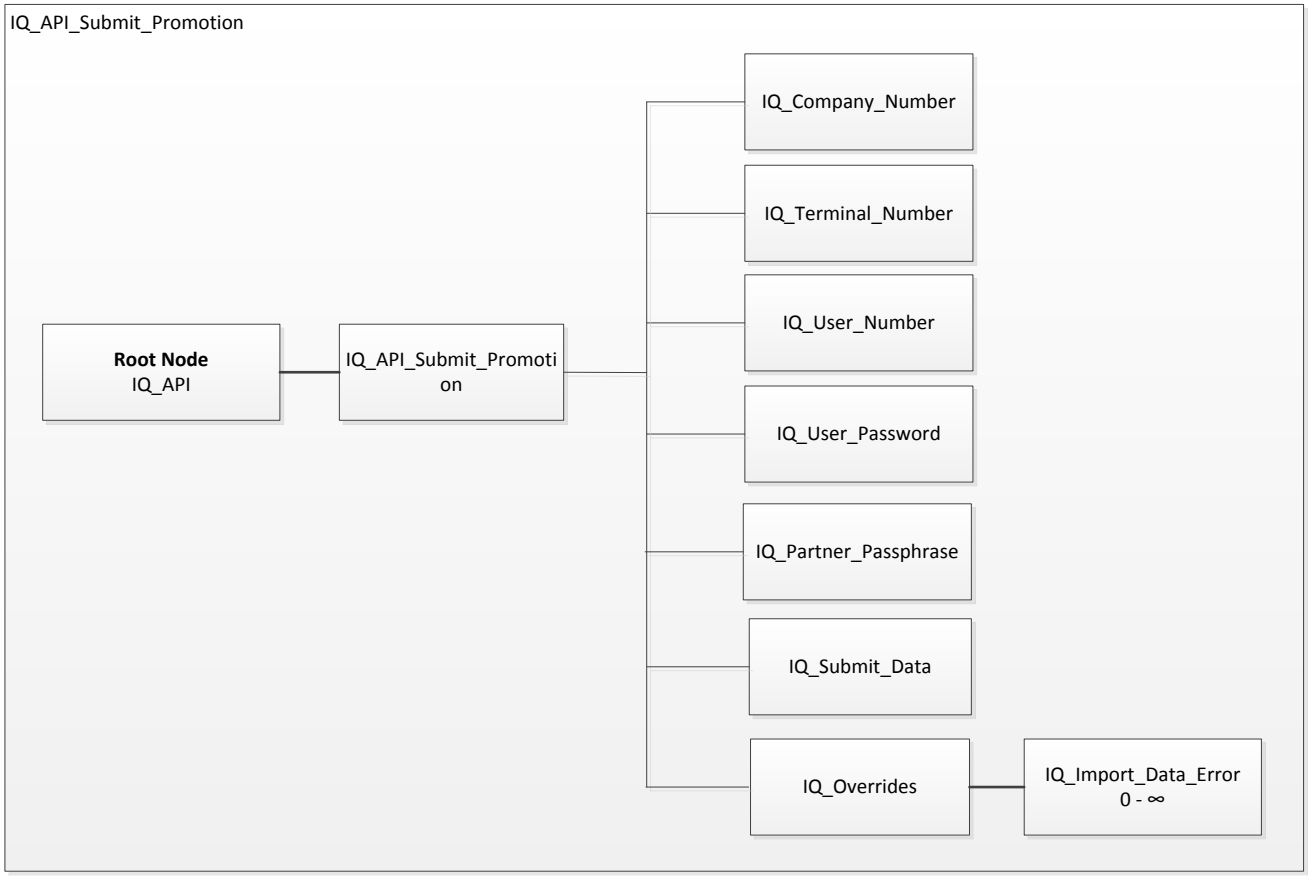
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_Promotion** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Stock Promotion related submissions. It contains the root node, the identifying node **<IQ_API_Submit_Promotion>**, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Promotion>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
```

```
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Promotion >
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnPromotionSubmitClick(Sender: TObject);
var
 LProc         : TIQ_EntAPI_Submit_Procedure;
 LFreeProc     : TIQ_EntAPI_Free_PChar;
 LFileName     : String;
 LXML          : TNativeXML;
 LXMLSource    : TNativeXML;
 LNewRoot      : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString    : PChar;
 LRes          : PChar;
 LXMLLength    : Integer;
 LResLength    : Integer;
 LResInt       : Integer;
 LPassword     : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Promotion ');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;
```

```
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnPromotionSubmit_Click(object sender, EventArgs e)
{
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Promotion (LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
   MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }
```

```
lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_SalesRep

Description: This method allows the host application to submit Sales Representative Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_SalesRep")] public static extern int IQ_API_Submit_SalesRep ([MarshalAs(UnmanagedType.LPStr)]string aParam, int
aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

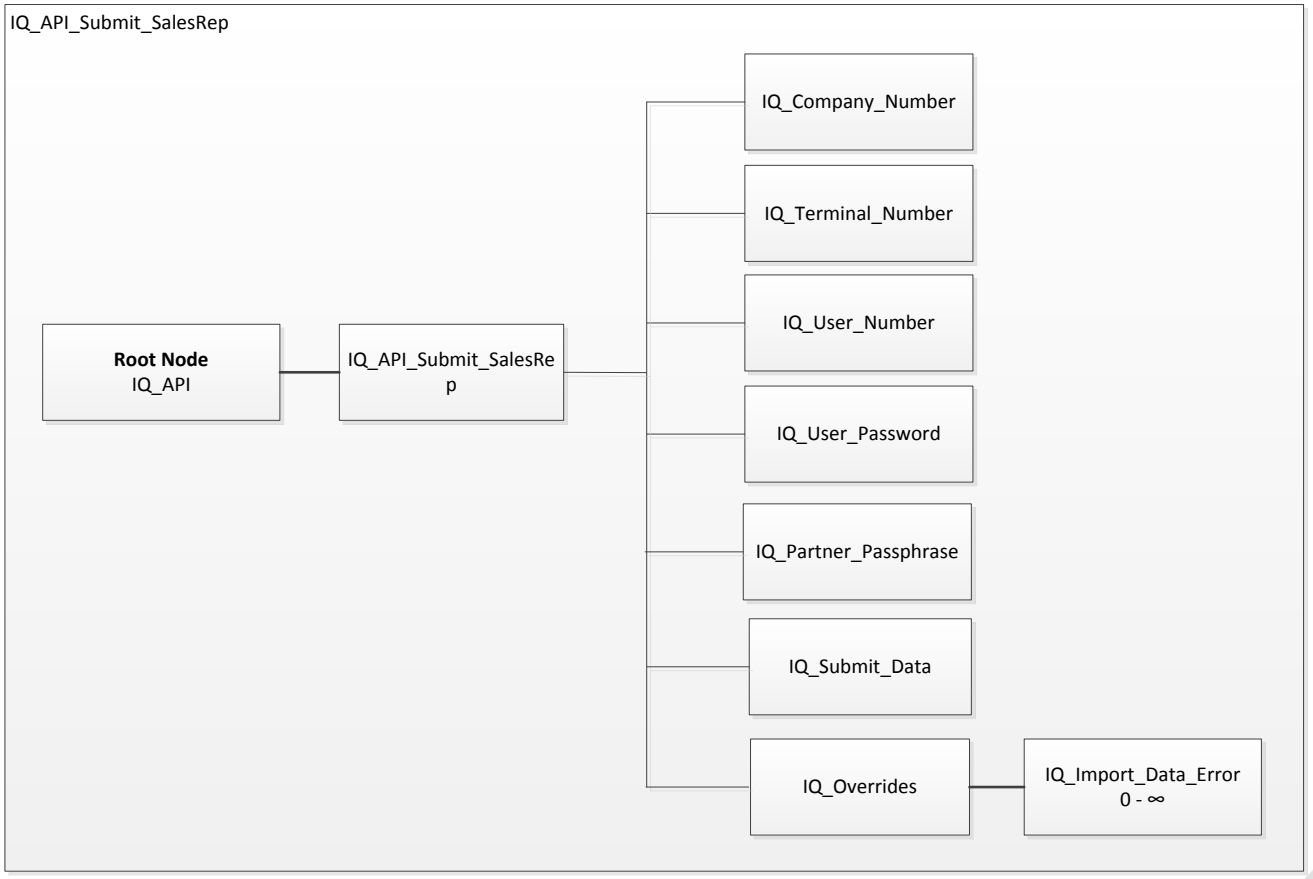
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the **IQ_API_Submit_SalesRep** node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software to which the provided data will be submitted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal that the API call will emulate.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user that the API call will emulate. Such user number will need to be enabled for API Access under the IQ Retail User Maintenance module.

IQ_User_Password: This is the IQ User's API Password (as set up under User Maintenance module). This password must be transmitted as a SHA1SUM Hash converted to Hexadecimal format and changed to upper case.

IQ_Partner_Passphrase: This value represents the SHA1SUM (converted to Hexadecimal format and changed to upper case) of a string value as determined by the IQ Partner. Such value will be interpreted by the API and, if recognized, be deemed as indication of an IQ Authorized API Partner. All API Calls are logged with such an indicator.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQMasterError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Sales Representative related submissions. It contains the root node, the identifying node **<IQ_API_Submit_SalesRep>**, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_SalesRep>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Partner_Passphrase>357E1068EB06668C2FC8C7AE713E4EA8D</IQ_Partner_Passphrase>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_SalesRep>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnSalesRepSubmitClick(Sender: TObject);
var
 LProc         : TIQ_EntAPI_Submit_Procedure;
 LFreeProc     : TIQ_EntAPI_Free_PChar;
 LFileName     : String;
 LXML          : TNativeXML;
 LXMLSource    : TNativeXML;
 LNewRoot      : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString    : PChar;
 LRes          : PChar;
 LXMLLength    : Integer;
 LResLength    : Integer;
 LResInt       : Integer;
 LPassword     : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
```

```
 try
  LProc     := GetProcAddress(FHandle, IQ_API_Submit_SalesRep');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;

  //..
            //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnSalesRepSubmit_Click(object sender, EventArgs e)
{
  //..
            //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;
```

```
LCallResult = IQ_API_Submit_SalesRep(LMessage, LMessageLength, out LResult, ref LResultLength);
LResultString = Marshal.PtrToStringAnsi(LResult);

if (LCallResult != 0)
{
  MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
}

lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
```

IQ_API_Submit_Debtor_Price_List

Description: This method allows the host application to submit Debtor Price List Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                 : TIQ_Type_Char_Param;
                                        aParam_Length         : TIQ_Type_Param_Length;
                                       Out aResult            : TIQ_Type_Char_Result;
                                        Var aResult_Length    : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint =
"IQ_API_Submit_Debtor_Price_List")] public static extern int IQ_API_Submit_Debtor_Price_List([MarshalAs(UnmanagedType.LPStr)]string aParam,
int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

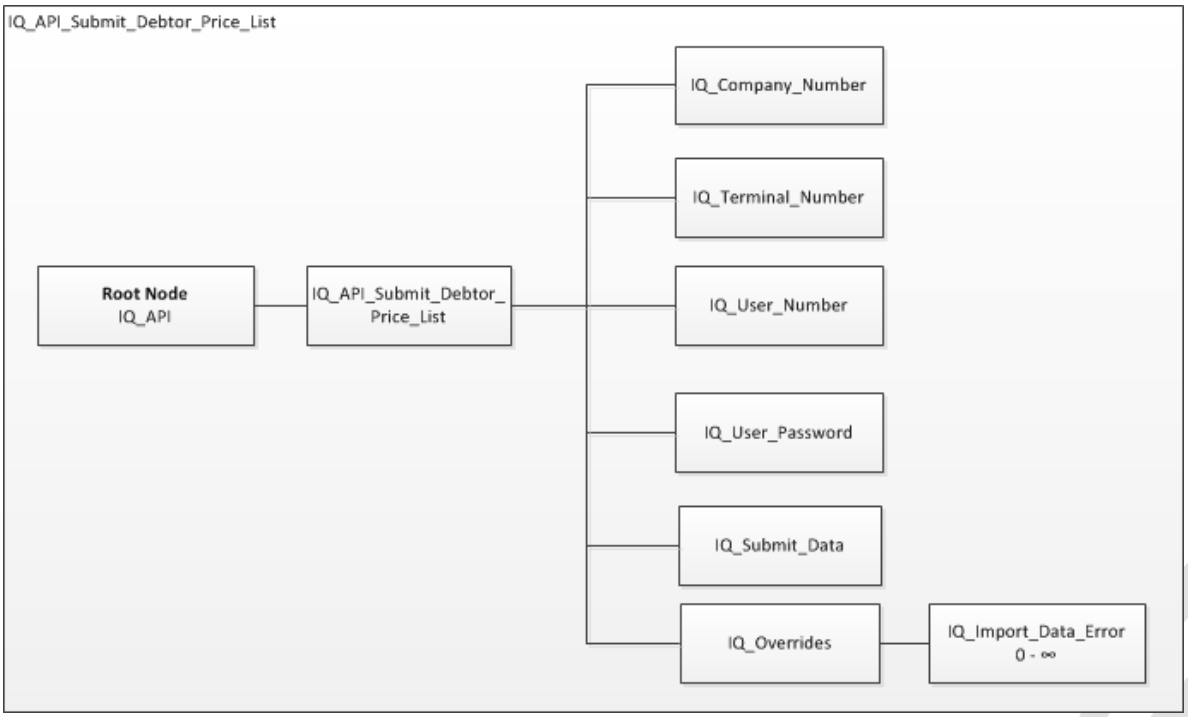
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Debtor_Price_List node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested debtor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal within the IQ Software from which the API is requesting information.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user within the IQ Software from which the API is requesting information.

IQ_User_Password: This is the password of the IQ User within the IQ Software. The password sent to the API needs to be hashed as well as converted to hex.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQJournalError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Debtor Price List related submissions. It contains the root node, the identifying node <IQ_API_Submit_Debtor_Price_List>, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
```

```
 <IQ_API_Submit_Debtor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Debtor_Price_List>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

*****See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnDebtorPriceListSubmitClick(Sender: TObject);
var
 LProc         : TIQ_EntAPI_Submit_Procedure;
 LFreeProc     : TIQ_EntAPI_Free_PChar;
 LFileName     : String;
 LXML          : TNativeXML;
 LXMLSource    : TNativeXML;
 LNewRoot      : TXMLNode;
 LSubmitDataNode : TXMLNode;
 LXMLString    : PChar;
 LRes          : PChar;
 LXMLLength    : Integer;
 LResLength    : Integer;
 LResInt       : Integer;
 LPassword     : String;
begin
 LPassword := 'Test';
 if not LoadDLL then Exit; // FHandle Outside of this Event
 try
  LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Debtor_Price_List');
  LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');
```

```
  if not Assigned(LProc) then Exit;
  if not Assigned(LFreeProc) then Exit;

  //..
        //Your code to generate the submission data according to XML Schema
  //..

  LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
  SetResult(Copy(LRes, 1, LResLength));
  LFreeProc(LRes);
 finally
  ReleaseDLL;
 end;
end;
```

Example Code [C#]:

```
private void btnDebtorPriceListSubmit_Click(object sender, EventArgs e)
{
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Debtor_Price_List(LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);
```

```
  if (LCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

IQ_API_Submit_Creditor_Price_List

Description: This method allows the host application to submit Creditor Price List Attributes to the IQ Database. This method expects 4 (four) parameters of which: The first is the XML formatted submission data, the second is the length of this information, the third is the result returned from the API DLL to the host application and the fourth is the length of this result returned. This method is a function and thus has a return type containing the last ERROR code in the event of an unsuccessful call of the method.

Type Declaration:

[DELPHI]

```
TIQ_EntAPI_Submit_Procedure  = Function(aParam                : TIQ_Type_Char_Param;
                                        aParam_Length        : TIQ_Type_Param_Length;
                                       Out aResult           : TIQ_Type_Char_Result;
                                        Var aResult_Length   : TIQ_Type_Param_Length): TIQ_Type_Result;
                stdcall;
```

[C#]

```
[DllImport(@"C:\iqelite\IQEnterprise4\Bin\IQEntAPI.DLL", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Ansi, EntryPoint = "IQ_API_Submit_Creditor_Price_List")] public static extern int IQ_API_Submit_Creditor_Price_List([MarshalAs(UnmanagedType.LPStr)]string aParam, int aParamLength, out IntPtr aResult, ref int aResultLength);
```

Input Parameters:

aParam : TIQ_Type_Char_Param – This is a parameter of type PChar and contains the XML formatted submission data.

aParam_Length : TIQ_Type_Param_Length – This parameter contains the length of **aParam**. This method will consider only characters within aParam from the first character up to the length specified in **aParam_Length**

Output Parameters:

aResult : TIQ_Type_Char_Result – This parameter is of type PChar and contains the XML formatted result of the method call. **.** Note the **out** keyword specifying that this parameter is intended for output (returning a result) only.

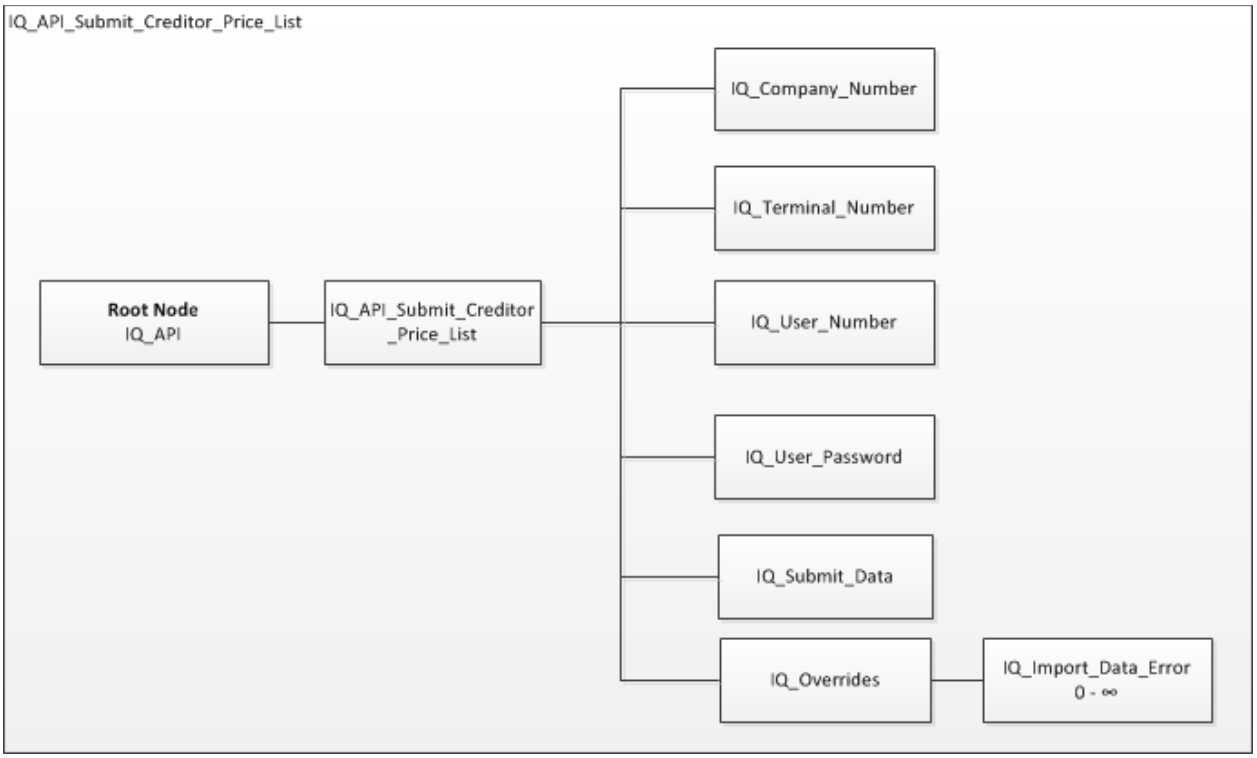
***** See [Memory Allocation] section for more details on handling PChar result parameters.**

aResult_Length : TIQ_Type_Param_Length – This parameter is of type Integer and is passed by value. This parameter will contain the length of aResult. This length can be used by the calling application to extract the relevant information from aResult.

Function Result: This method is a function and returns, as its result, the last Error Code (in the event of an unsuccessful call to this method). This value is of type TIQ_Type_Result (an Integer value). See global type declarations for error code declarations.

Format of XML Submission:

The XML formatted submission is generated by the host application and contains information specifying the IQ Company to which the data will be submitted, relevant security override information and the actual submission data.



This request type is identified by the IQ_API_Submit_Creditor_Price_List node (a child node of the, by now, well known IQ_API node).

IQ_Company_Number: This is the IQ Company identifying number (eg. 001, ABC, TES) that indicates the company within the IQ Software from which the requested creditor information should be extracted.

IQ_Terminal_Number: This is the IQ Terminal identifying number (eg. 1, 10, 12) that indicates the terminal within the IQ Software from which the API is requesting information.

IQ_User_Number: This is the IQ User (Staff) identifying number (eg. 1, 10, 12) that indicates the user within the IQ Software from which the API is requesting information.

IQ_User_Password: This is the password of the IQ User within the IQ Software. The password sent to the API needs to be hashed as well as converted to hex.

IQ_Submit_Data: This node contains the actual submission data in XML format as per the IQ XML Schema documents. The root node of this data will be <IQ_Root_XML>.

IQ_Overrides: Due to the security structure within the IQ Enterprise family of software there are certain security overrides that will need to be approved by the host application and submitted as part of the submission request. These security overrides will determine the successful importing of XML documents. If these overrides are not approved, the submission may fail with relevant errors. The IQ_Overrides node will contain zero (0) to many IQ_Import_Data_Error nodes.

IQ_Import_Data_Error: This node will contain the enumerated error value as per Enumerated Definition Types for TIQJournalError. The value will be automatically converted to the native system type and will be used to determine which security items have been approved by the host application. These security overrides will determine which security characteristics (related to the specific import) are to be considered allowable vs not allowable.

XML Request Example:

This example shows the typical format of submission data for Creditor Price List related submissions. It contains the root node, the identifying node <IQ_API_Submit_Creditor_Price_List>, the relevant Company Number, Terminal Number, User Number, User Password, the actual submission data and no overrides. This example contains no submission data (mainly due to the size / number of lines which would clutter this document). The IQ_Submit_Data would contain the complete set of submission data in an actual request).

```
<?xml version="1.0" encoding="utf-16"?>
<IQ_API>
 <IQ_API_Submit_Creditor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>1</IQ_User_Number>
  <IQ_User_Password>357E1068EB06123C2FC8C7AE713E4EA8D</IQ_User_Password>
  <IQ_Submit_Data />
  <IQ_Overrides />
 </IQ_API_Submit_Creditor_Price_List>
</IQ_API>
```

Format of XML Response:

The format of the XML Response will contain the IQ_API_Error node.

***See XML Formatted Response section for more details.**

Example Code [DELPHI]:

```
procedure TfrmAPITest.btnCreditorPriceListSubmitClick(Sender: TObject);
var
 LProc        : TIQ_EntAPI_Submit_Procedure;
```

```
  LFreeProc      : TIQ_EntAPI_Free_PChar;
  LFileName      : String;
  LXML           : TNativeXML;
  LXMLSource     : TNativeXML;
  LNewRoot       : TXMLNode;
  LSubmitDataNode : TXMLNode;
  LXMLString     : PChar;
  LRes           : PChar;
  LXMLLength     : Integer;
  LResLength     : Integer;
  LResInt        : Integer;
  LPassword      : String;
begin
  LPassword := 'Test';
  if not LoadDLL then Exit; // FHandle Outside of this Event
  try
    LProc    := GetProcAddress(FHandle,'IQ_API_Submit_Creditor_Price_List');
    LFreeProc := GetProcAddress(FHandle, 'IQ_API_Free_PChar');

    if not Assigned(LProc) then Exit;
    if not Assigned(LFreeProc) then Exit;

    //..
            //Your code to generate the submission data according to XML Schema
    //..

    LResInt := LProc(LXMLString, LXMLLength, LRes, LResLength);
    SetResult(Copy(LRes, 1, LResLength));
    LFreeProc(LRes);
  finally
    ReleaseDLL;
  end;
end;
```

Example Code [C#]:

```
private void btnCreditorPriceListSubmit_Click(object sender, EventArgs e)
```

```
{
  //..
        //Your code to generate the submission data according to XML Schema
  //..

  string LMessage;
  int LMessageLength;
  IntPtr LResult;
  string LResultString;
  int LResultLength = 0;
  int LCallResult;

  LMessage = LStringWriter.ToString();
  LMessageLength = LMessage.Length;

  LCallResult = IQ_API_Submit_Creditor_Price_List(LMessage, LMessageLength, out LResult, ref LResultLength);
  LResultString = Marshal.PtrToStringAnsi(LResult);

  if (LCallResult != 0)
  {
    MessageBox.Show("An Error Occurred. Error Code [" + LCallResult.ToString() + "]");
  }

  lstResult.Text = FormatXML(LResultString.Substring(0, LResultLength));
}
```

***** End of Exposed / Available Calls *****

Appendix

IQ XML Schemas
-See Separate Documentation-

API REST Server Setup

The REST Server is available in your IQ Enterprise installation folder. The filename is called "IQEntAPIRestServer.exe", this executable serves as both the setup application and the windows service. If the executable is run as administrator you will be presented with the IQ main window.

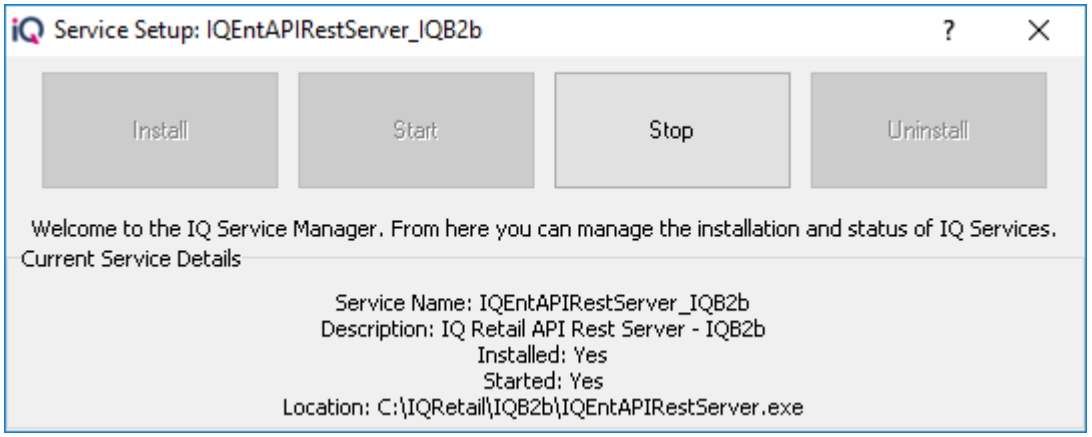
Login with administrator credentials.

Menu's available:

- ***Setup***
  - **Service**
    The Service Setup window will allow the user to Install, Start, Stop and Uninstall the API REST Server Service. (Fig.1 below)
  - **Web Service Configuration**
    This window will allow you to setup various configurations for the REST Server Service. (Fig.2 below)
    - Port
    - IP Address
    - Enable Authentication
    - Authentication Scheme
    - Enable SSL / TLS
    - Certificate File
    - Key File
    - Root Certifate File
- ***Windows***
  - **Cascade**
  - **Tile**
  - **Arrange Icons**
  - **Minimize All**

* Minimum require setup will only be the port number to use.

Fig.1 - Service Seteup screen



Commands (CMD)

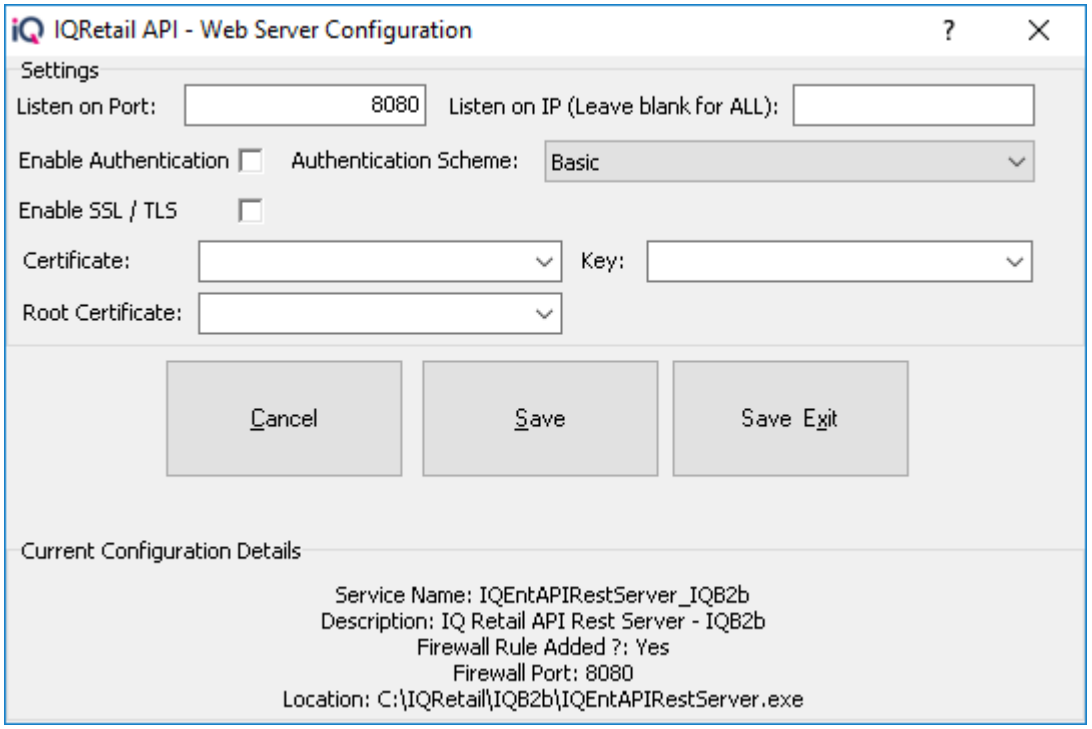
IQEntAPIRestServer.exe -install

IQEntAPIRestServer.exe -uninstall

IQEntAPIRestServer.exe -start

IQEntAPIRestServer.exe -stop

Fig.2 - Web Server Configuration screen

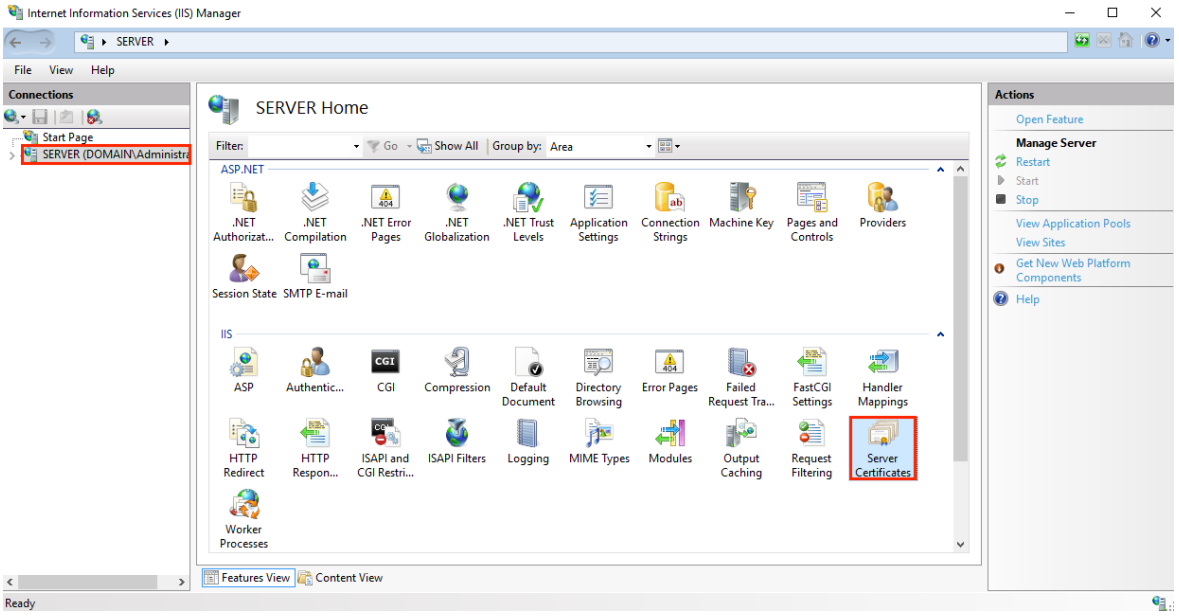


Creating a CSR and installing your SSL certificate on your Windows server 2016

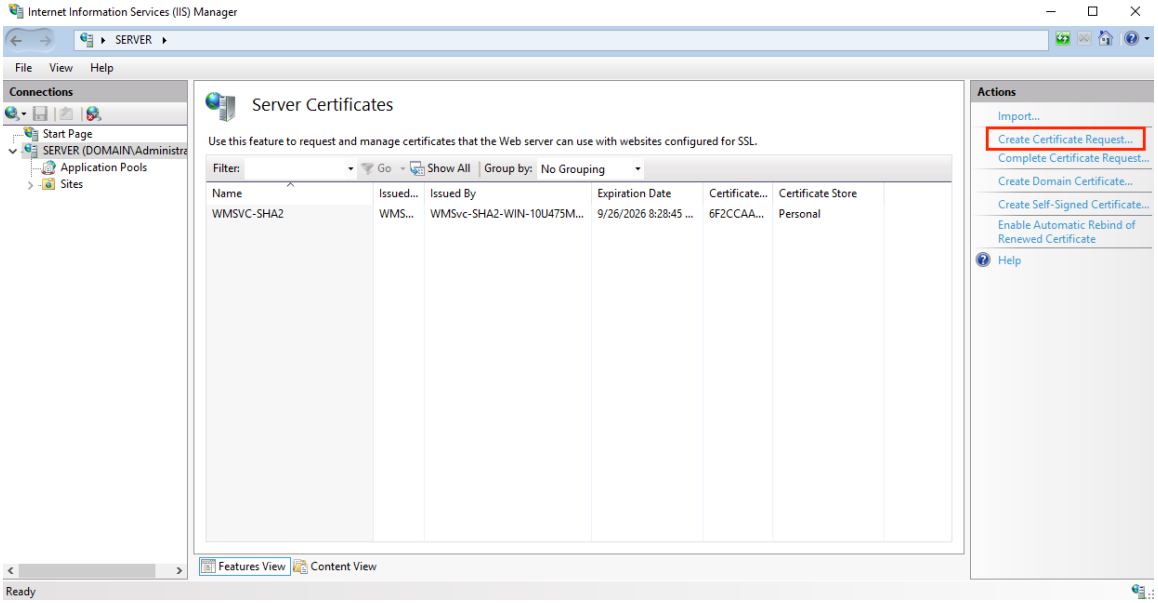
IIS 10: How to Create Your CSR on Windows Server 2016

Using IIS 10 to Create Your CSR

1. In the **Windows** start menu, **type Internet Information Services (IIS) Manager** and open it.
2. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), locate and click the server name.

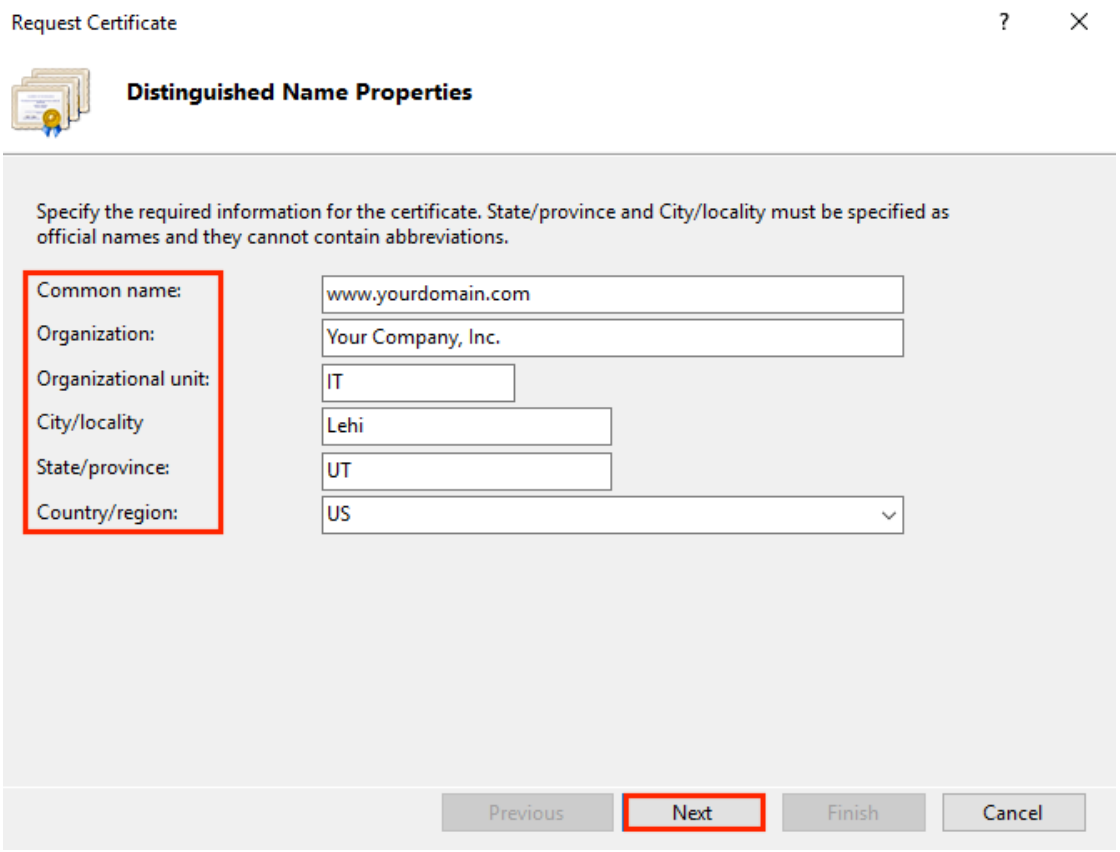


3. On the server name **Home** page (center pane), in the **IIS** section, double-click **Server Certificates**.
4. On the **Server Certificates** page (center pane), in the **Actions** menu (right pane), click the **Create Certificate Request…** link.



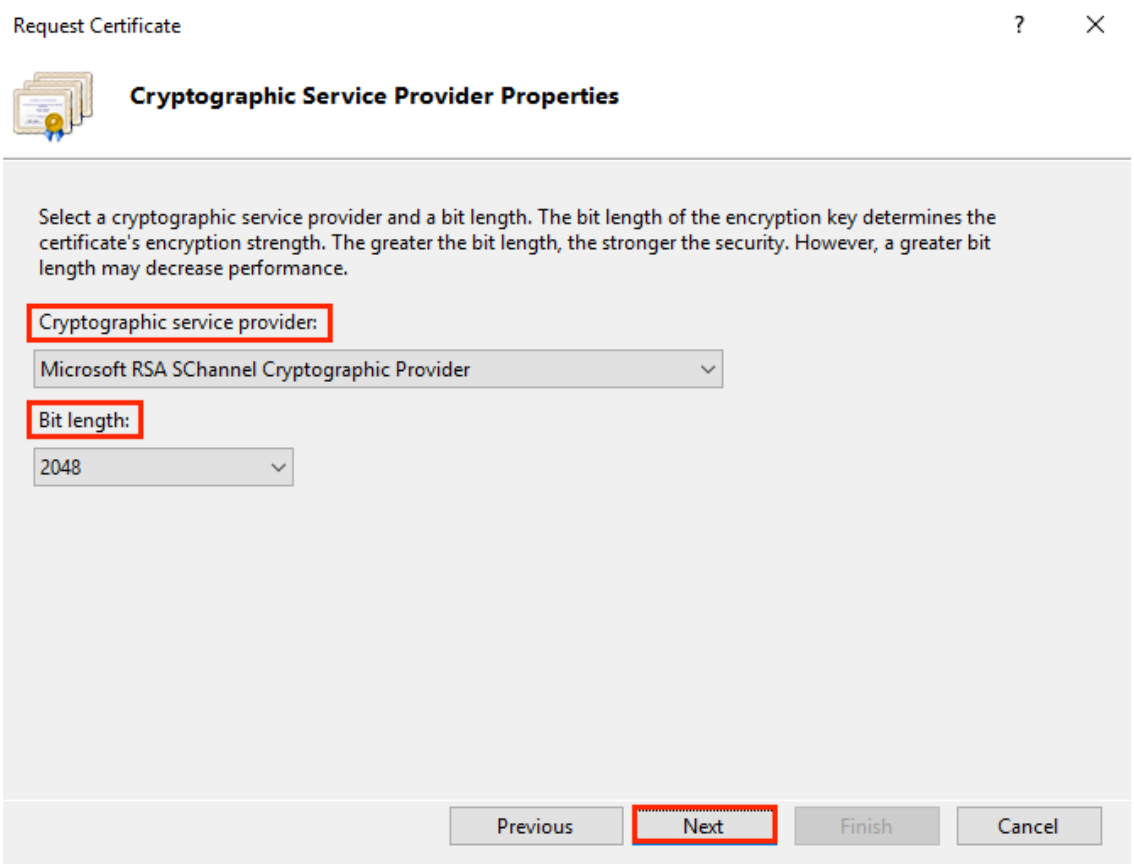
5. In the **Request Certificate** wizard, on the **Distinguished Name Properties** page, provide the information specified below and then click **Next**:

Common name:	Type the fully-qualified domain name (FQDN) (e.g., *www.example.com*).
Organization:	Type your company's legally registered name (e.g., *YourCompany, Inc.*).
Organizational unit:	The name of your department within the organization. Frequently this entry will be listed as "IT", "Web Security," or is simply left blank.
City/locality:	Type the city where your company is legally located.
State/province:	Type the state/province where your company is legally located.
Country:	In the drop-down list, select the country where your company is legally located.



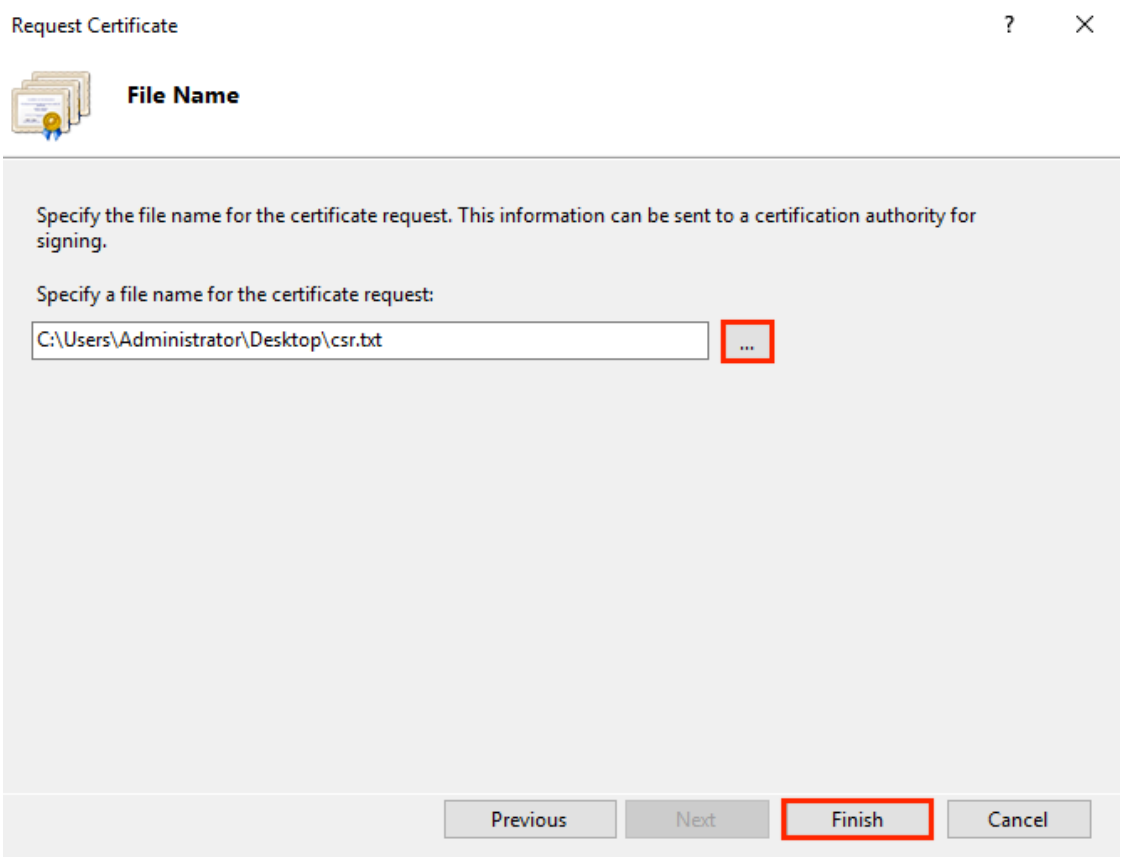
6. On the **Cryptographic Service Provider Properties** page, provide the information below and then click **Next**.

Cryptographic service provider:	In the drop-down list, select **Microsoft RSA SChannel Cryptographic Provider**, unless you have a specific cryptographic provider.
Bit length:	In the drop-down list select **2048**, unless you have a specific reason for opting for larger bit length.



7. On the **File Name** page, under **Specify a file name for the certificate request**, click the **…** box to browse to a location where you want to save your CSR.

   Note: Remember the filename that you choose and the location to which you save your csr.txt file. If you just enter a filename without browsing to a location, your CSR will end up in C:\Windows\System32.



8. When you are done, click **Finish**.
9. Use a text editor (such as Notepad) to open the file. Then, copy the text, including the **-----BEGIN NEW CERTIFICATE REQUEST-----** and **-----END NEW CERTIFICATE REQUEST-----** tags, and paste it into the DigiCert order form.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICvDCCAaQCAQAwdzELMAkGA1UEBhMCVVMxEjAQBgNVBAgTCVlvdXJTdGF0ZTER
MA8GA1UEBxMIWW91ckNpdHkxCzAJBgNVBAsTAklUMRowGAYDVQQKExFZb3VyQ29t
cGFueSwgSW5jLjEYMBYGA1UEAxMPd3d3LmV4YW1wbGUuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA379BFFxfACdXsUk2wrQka/nAlKbo+I9DAW32
+/SRxj/KtXVddscKW1obHGpMKPw4meJqOpQwJkIChYjSUQSpPKzdGpccDMf/eoF0
J7EaQ2szLv9AqdRQw2Aaek8SmocVmd3LxEOX4VvALBOMLHVrB5/vhYfGECLJbc3l
RdEbdXyHDtHklRAoIVQCfjTwBWGNAD337vmHW7Q0R6FYUoa4fcJh7Rv6jHSywqwx
7pVfaDb2PuTgUhw7wksKNFxccG0xcTMr/+GrciHEuZ0chq86CBP9RIyLpp2+RMSf
m6rMEYm9o65j7vEYaKEJUOJtA5MIs/ZjaXfS1LjXurLU0nCOQQIDAQABoAAwDQYJ
KoZIhvcNAQEFBQADggEBAK159goyAYOpcnrQ2EvCGlizrK1kS3D8JjnAiP1NHrjB
/qdTYR+/8Dr/hMcwwU5ThGAVf68eMkk6tUNwAdpZ9C904Js2z+ENEbO8GA0Fc4rw
ix7vb15vSXe3shGijRGIzzHVGRoR3r7xQtIuMaDAr3xlV8jHbcvZTcpX0Kbq6H1G
NLA4CXsOI4KGwu4FXfSzJEGb3gEJD8HaMP8V8er5G0owv/g/9Z/1/b0g97kAcUwk
M2eDsvPhMx/pENGbnLPe4XMy7NPiEdzFnaYtUy2BDcXj3ZQEWxRWklERgg9/YcWI
obf5ziuNm1Df24NBt5tpCNzfGviKT6/RYfWg3dMaKxc=
-----END NEW CERTIFICATE REQUEST-----
```

10. After you receive your SSL certificate from DigiCert, you can install it.

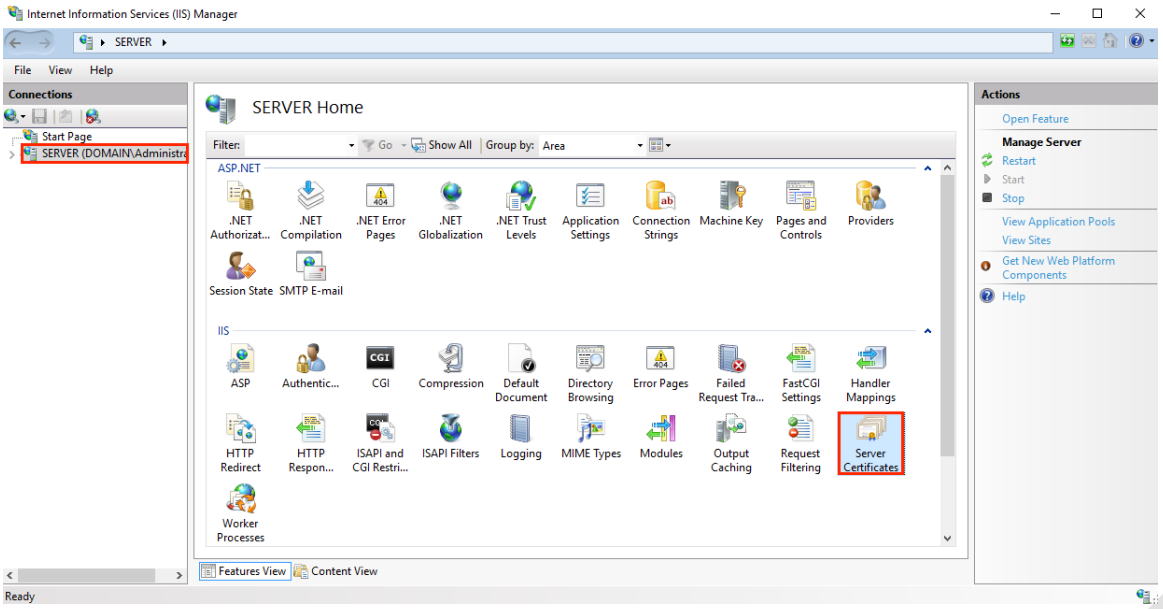
IIS 10: How to Install and Configure Your SSL Certificate on Windows Server 2016

After we validate and issue your SSL certificate, you need to install it on the Windows 2016 server where the CSR was generated.

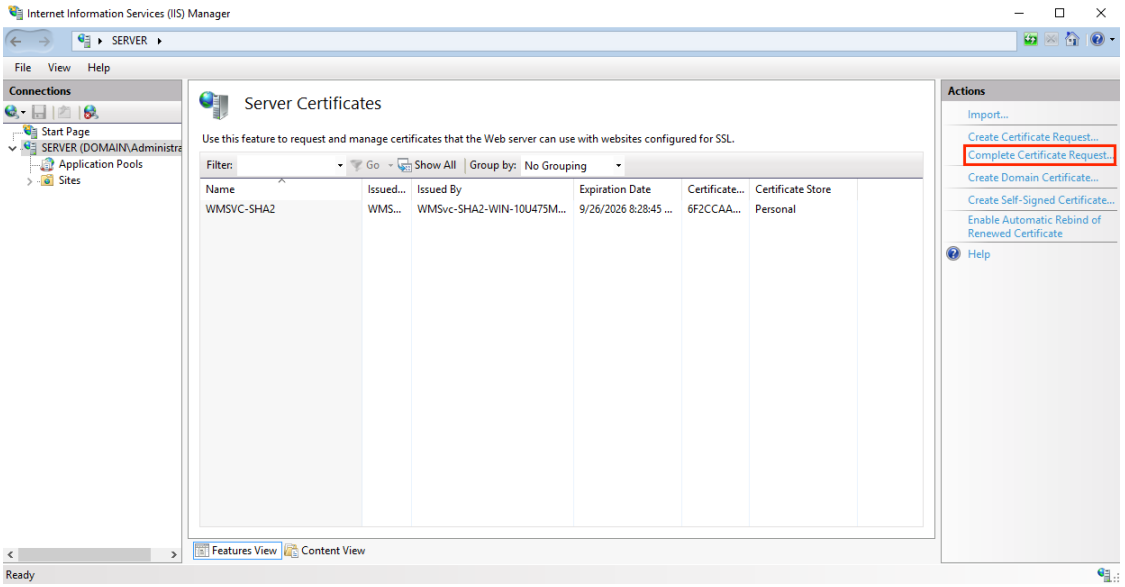
(Single Certificate) How to install your SSL certificate and configure the server to use it

Install SSL Certificate

1. On the server where you created the CSR, save the SSL certificate .cer file (e.g., *your_domain_com.cer*) that DigiCert sent to you.
2. In the **Windows** start menu, type ***Internet Information Services (IIS) Manager*** and open it.
3. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), locate and click the server name.

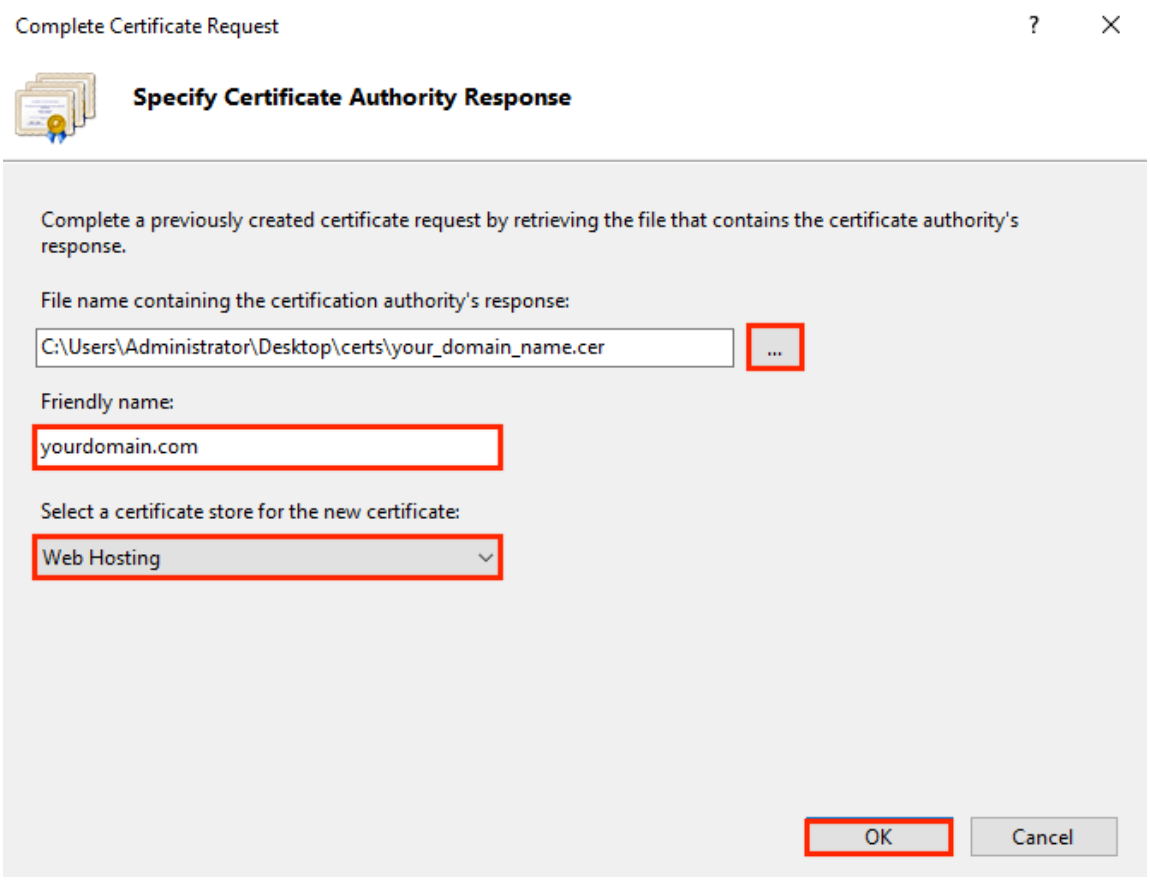


4. On the server name **Home** page (center pane), in the **IIS** section, double-click **Server Certificates**.
5. On the **Server Certificates** page (center pane), in the **Actions** menu (right pane), click the **Complete Certificate Request…** link.



6. In the **Complete Certificate Request** wizard, on the **Specify Certificate Authority Response** page, do the following and then click **OK**:

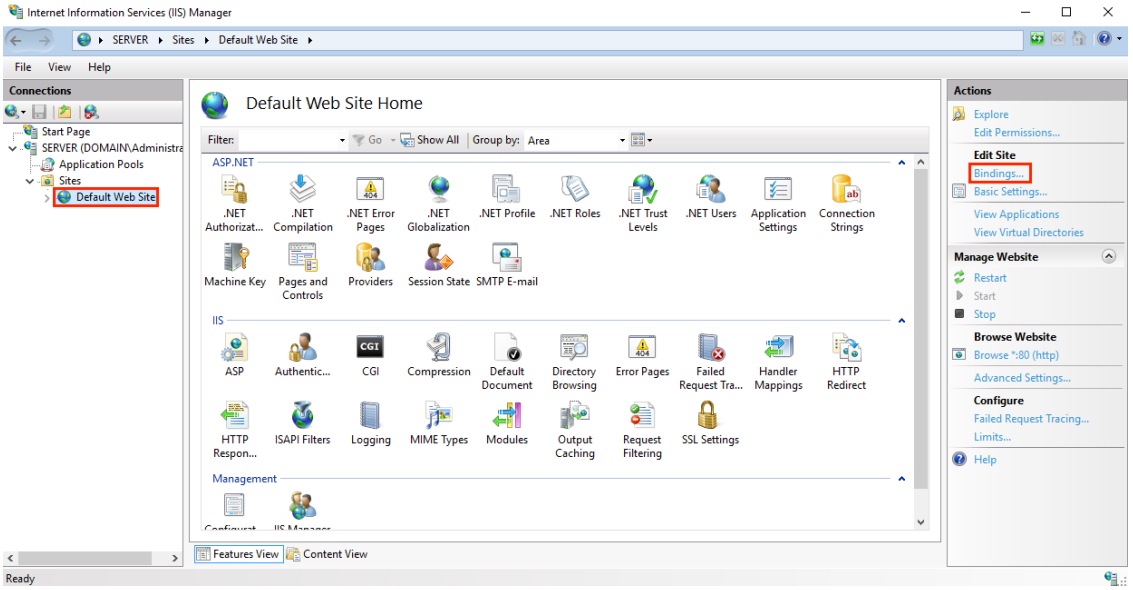
File name containing the certificate authority's response:	Click the **…** box and browse to and select the .cer file (e.g., *your_domain_com.cer*) that DigiCert sent to you.
Friendly name:	Type a friendly name for the certificate. The friendly name is not part of the certificate; instead, it is used to identify the certificate. We recommend that you add DigiCert and the expiration date to the end of your friendly name, for example: yoursite-digicert-(expiration date). This information helps identify the issuer and expiration date for each certificate. It also helps distinguish multiple certificates with the same domain name.
Select a certificate store for the new certificate:	In the drop-down list, select **Web Hosting**.



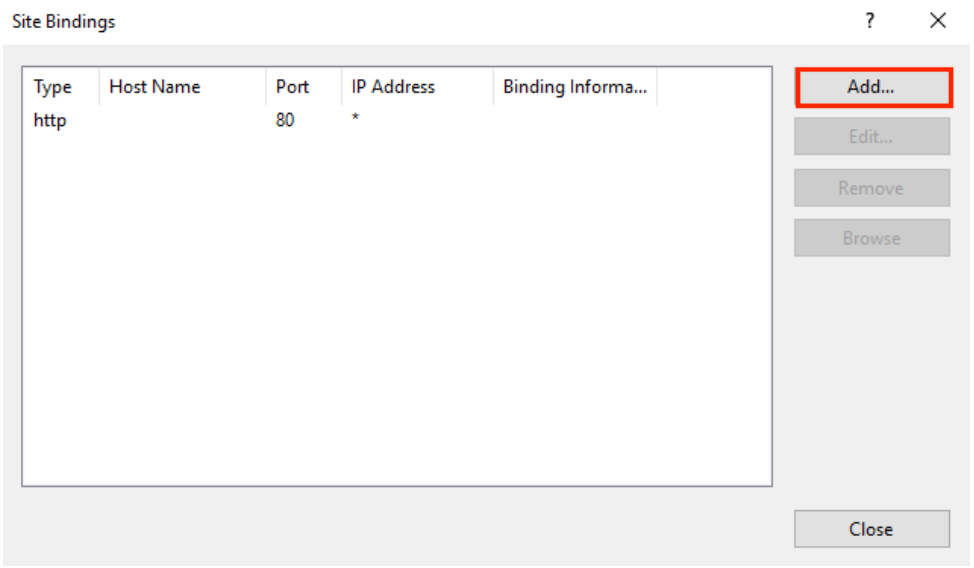
7. Now that you've successfully installed your SSL certificate, you need to assign the certificate to the appropriate site.

Assign SSL Certificate

8. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), expand the name of the server on which the certificate was installed. Then expand **Sites** and click the site you want to use the SSL certificate to secure.



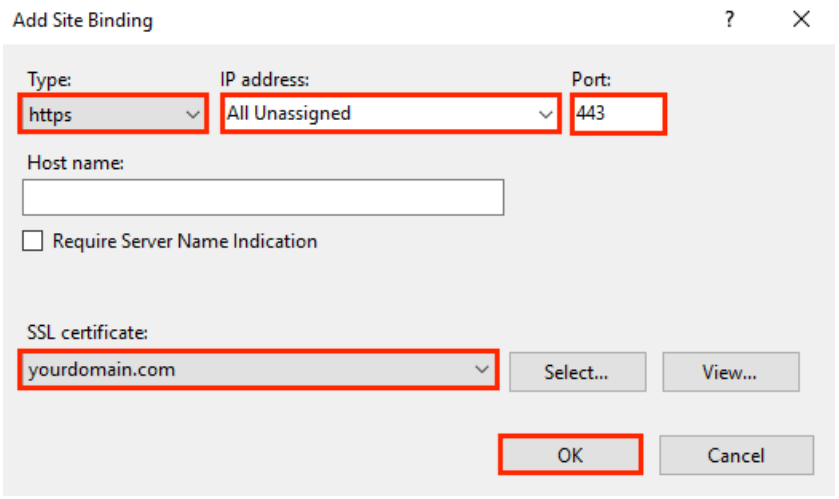
9. On the website **Home** page, in the **Actions** menu (right pane), under **Edit Site**, click the **Bindings…** link.
10. In the **Site Bindings** window, click **Add**.



11. In the **Add Site Bindings** window, do the following and then click **OK**:

Type:	In the drop-down list, select **https**.
IP address:	In the drop-down list, select the IP address of the site or select **All Unassigned**.
Port:	Type port ***443***. The port over which traffic is secure by SSL is port 443.

SSL certificate: In the drop-down list, select your new SSL certificate (e.g., *yourdomain.com*).



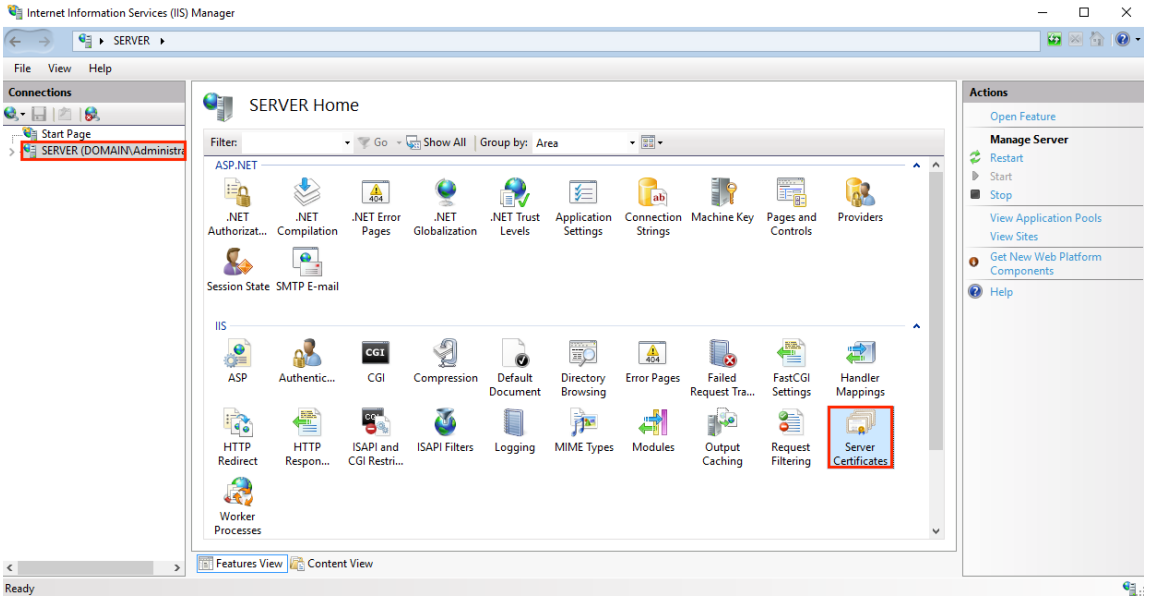
12. Your SSL certificate is now installed, and the website configured to accept secure connections.

(Multiple Certificates) How to install your SSL certificates and configure the server to use them using SNI

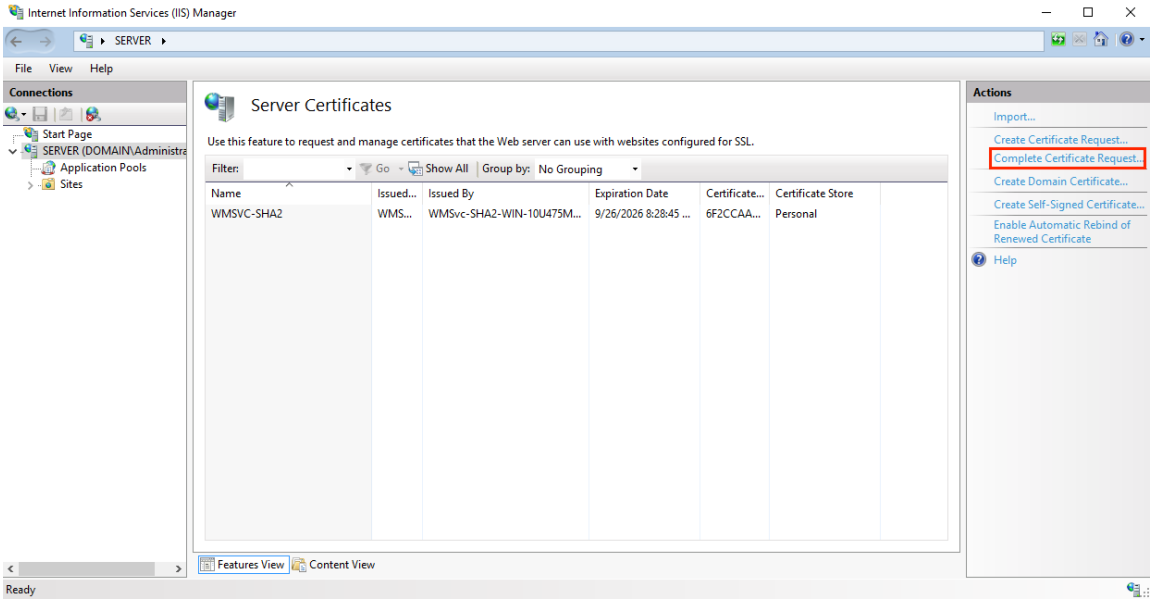
Install First SSL Certificate

Do this first set of instructions only once, for the first SSL certificate.

1. On the server where you created the CSR, save the SSL certificate .cer file (e.g., *your_domain_com.cer*) that DigiCert sent to you.
2. In the **Windows** start menu, type ***Internet Information Services (IIS) Manager*** and open it.
3. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), locate and click the server name.



4. On the server name **Home** page (center pane), in the **IIS** section, double-click **Server Certificates**.
5. On the **Server Certificates** page (center pane), in the **Actions** menu (right pane), click the **Complete Certificate Request…** link.

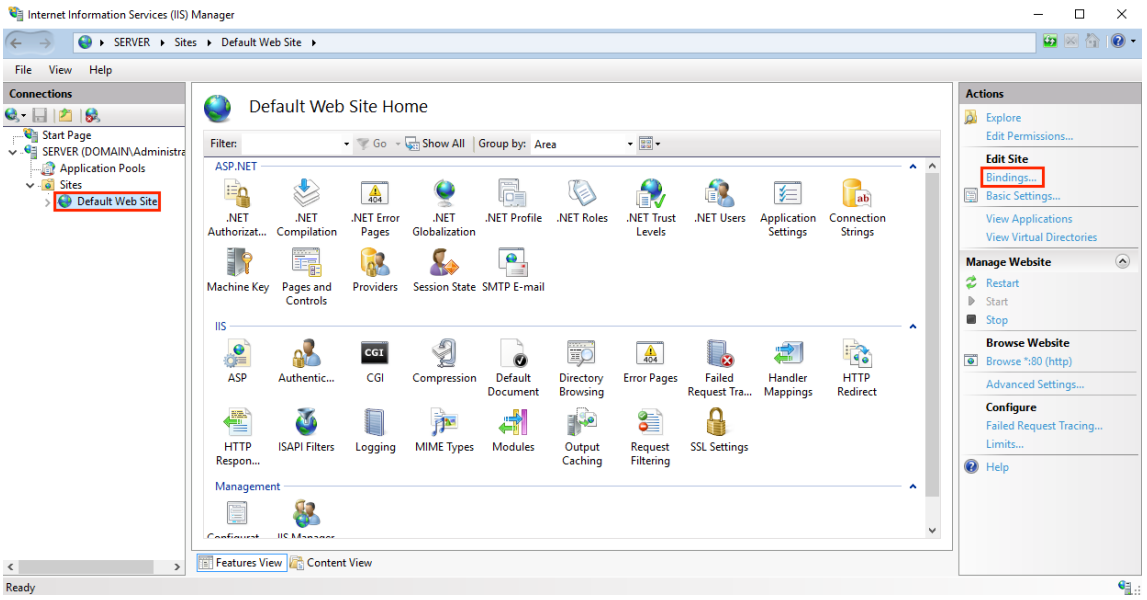


6. In the **Complete Certificate Request** wizard, on the **Specify Certificate Authority Response** page, do the following and then click **OK**:

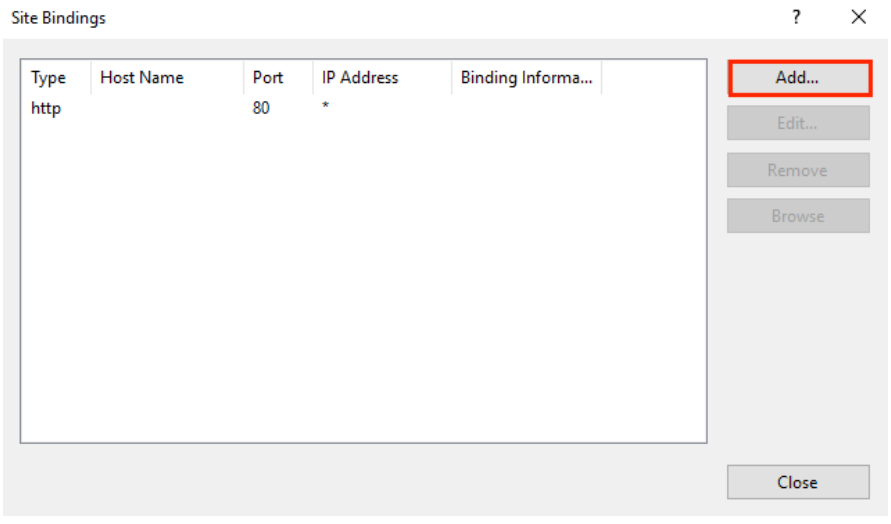
File name containing the	Click the **…** box and browse to and select the .cer file

certificate authority's response:	(e.g., *your_domain_com.cer*) that DigiCert sent to you.
Friendly name:	Type a friendly name for the certificate. The friendly name is not part of the certificate; instead, it is used to identify the certificate. We recommend that you add DigiCert and the expiration date to the end of your friendly name, for example: yoursite-digicert-(expiration date). This information helps identify the issuer and expiration date for each certificate. It also helps distinguish multiple certificates with the same domain name.
Select a certificate store for the new certificate:	In the drop-down list, select **Web Hosting**.

7. Now that you've successfully installed your SSL certificate, you need to assign the certificate to the appropriate site.
8. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), expand the name of the server on which the certificate was installed. Then expand **Sites** and click the site you want to use the SSL certificate to secure.



9. On the website **Home** page, in the **Actions** menu (right pane), under **Edit Site**, click the **Bindings…** link.
10. In the **Site Bindings** window, click **Add**.



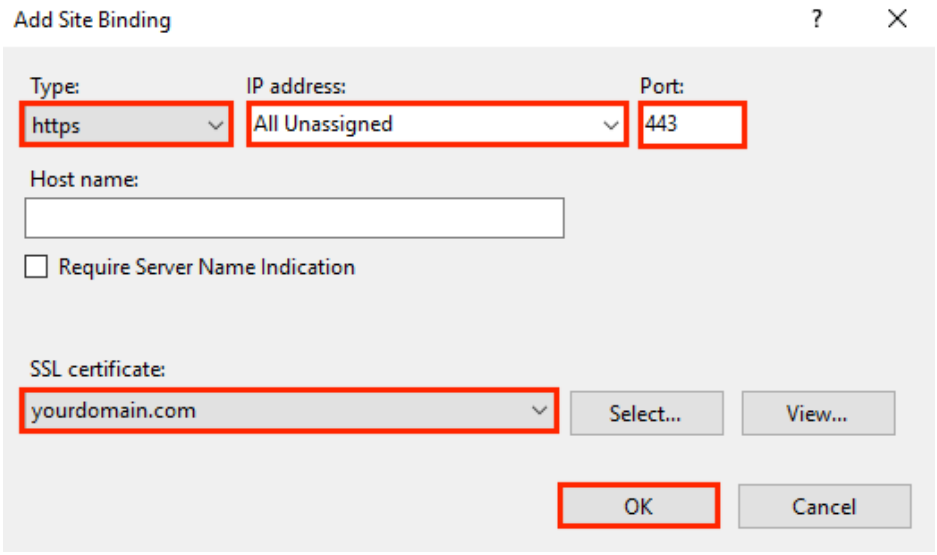
11. In the **Add Site Bindings** window, do the following and then click **OK**:

Type: In the drop-down list, select **https**.

IP address: In the drop-down list, select the IP address of the site or select **All Unassigned**.

Port: Type port ***443***. The port over which traffic is secure by SSL is port 443.

SSL certificate: In the drop-down list, select your new SSL certificate (e.g., *yourdomain.com*).

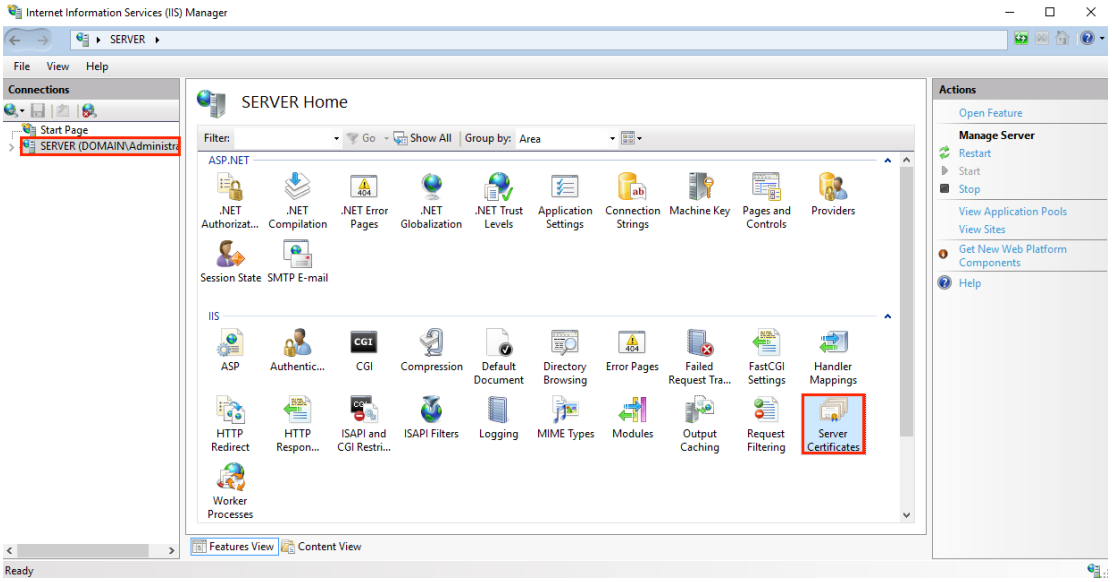


12. Your first SSL certificate is now installed, and the website configured to accept secure connections.

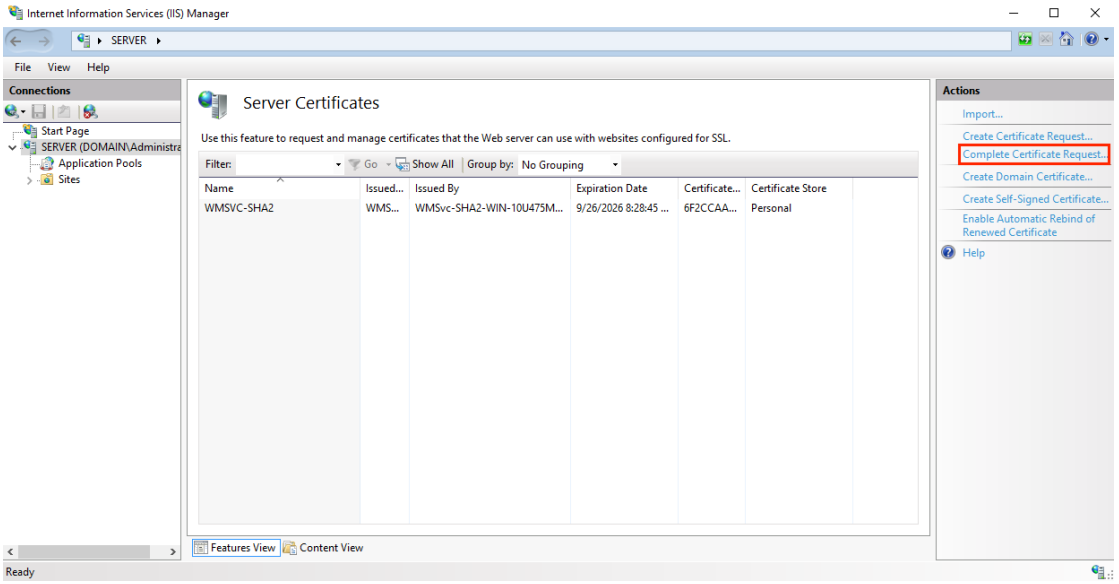
Install Additional SSL Certificates

To install and assign each additional SSL certificate, repeat the steps below, as needed.

1. On the server where you created the CSR, save the SSL certificate .cer file (e.g., *your_domain_com.cer*) that DigiCert sent to you.
2. In the **Windows** start menu, type ***Internet Information Services (IIS) Manager*** and open it.
3. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), locate and click the server name.



4. On the server name **Home** page (center pane), in the **IIS** section, double-click **Server Certificates**.
5. On the **Server Certificates** page (center pane), in the **Actions** menu (right pane), click the **Complete Certificate Request…** link.



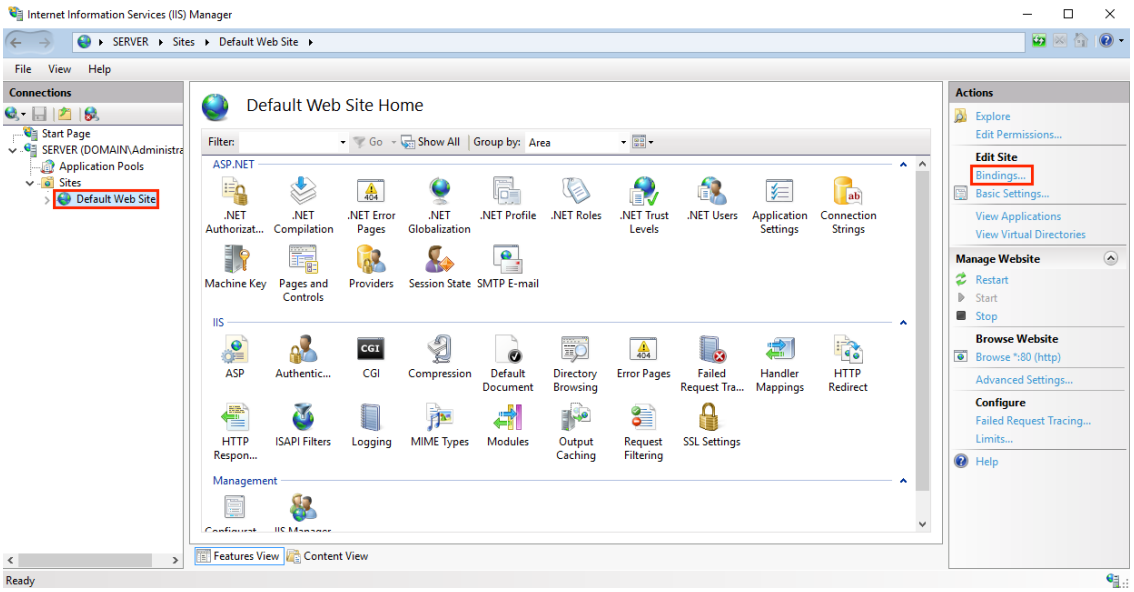
6. In the **Complete Certificate Request** wizard, on the **Specify Certificate Authority Response** page, do the following and then click **OK**:

File name containing the	Click the **…** box and browse to and select the .cer file

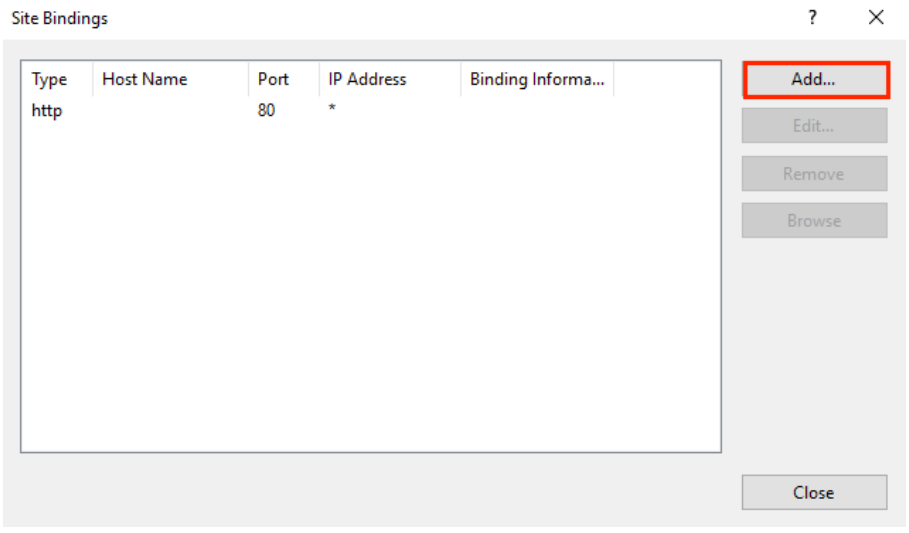
certificate authority's response:	(e.g., *your_domain_com.cer*) that DigiCert sent to you.
Friendly name:	Type a friendly name for the certificate. The friendly name is not part of the certificate; instead, it is used to identify the certificate. We recommend that you add DigiCert and the expiration date to the end of your friendly name, for example: yoursite-digicert-(expiration date). This information helps identify the issuer and expiration date for each certificate. It also helps distinguish multiple certificates with the same domain name.
Select a certificate store for the new certificate:	In the drop-down list, select **Web Hosting**.

7. Now that you've successfully installed your SSL certificate, you need to assign the certificate to the appropriate site.
8. In **Internet Information Services (IIS) Manager**, in the **Connections** menu tree (left pane), expand the name of the server on which the certificate was installed. Then expand **Sites** and click the site you want

to use the SSL certificate to secure.



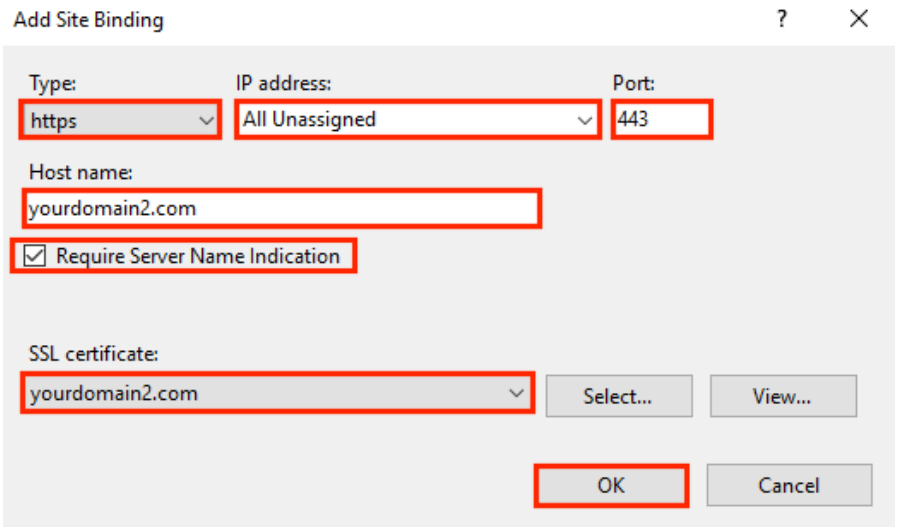
9. On the website Home page, in the Actions menu (right pane), under Edit Site, click the Bindings… link.
10. In the **Site Bindings** window, click **Add**.



11. In the **Add Site Bindings** window, do the following and then click **OK**:

Type: In the drop-down list, select **https**.

IP address:	In the drop-down list, select the IP address of the site or select **All Unassigned**.
Port:	Type port ***443***. The port over which traffic is secure by SSL is port 443.
Host name:	Type the host name that you want to secure.
Require Server Name Indication:	After you enter the host name, check this box. This is required for all additional certificates/sites, after you've installed the first certificate and secured the primary site.
SSL certificate:	In the drop-down list, select an additional SSL certificate (e.g., *yourdomain2.com*).



12. You have successfully installed another SSL certificate and configured the website to accept secure connections.

XML & JSON Examples

IQ_API_Request_GenericSQL

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<IQ_API>
<IQ_API_Request_GenericSQL>
<IQ_Company_Number>001</IQ_Company_Number>
<IQ_Terminal_Number>1</IQ_Terminal_Number>
<IQ_User_Number>99</IQ_User_Number>
<IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
<IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
<IQ_SQL_Text>Select * From Stock Top 10;</IQ_SQL_Text>
</IQ_API_Request_GenericSQL>
</IQ_API>
```

```
{
	"IQ_API": {
		"IQ_API_Request_GenericSQL": {
			"IQ_Company_Number": "001",
			"IQ_Terminal_Number": 1,
			"IQ_User_Number": 99,
			"IQ_User_Password": "C89ECE949D7A657B4508788FD2496088EABFD870",
			"IQ_Partner_Passphrase": "743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA",
			"IQ_SQL_Text": "Select * From Stock Top 10;"
		}
	}
}
```

Response

```
{
	"IQ_API_Error": [{
		"IQ_Error_Code": 0
	}],
	"IQ_API_Result_Data": {
		"records": [{
			"CODE": "123123",
			"BARCODE": "123123",
			"GENCODE": "",
			"DESCRIPT": "",
			"ALT_DESCRIPT": "",
			"SUPPLIERCO": "",
			"SINGLE_SER": "",
			"DEPARTMENT": "001",
		}]
```

```
            }
    }
```

Zero Records Result

```
<?xml version="1.0" encoding="UTF-8"?>
<IQ_API_Result>
   <IQ_API_Error>
      <IQ_Error_Code>5</IQ_Error_Code>
      <IQ_Error_Description>Record Count Zero. No Records To
                            Retrieve</IQ_Error_Description>
      <IQ_Error_Data>
        <IQ_Error_Data_Item>
          <IQ_Error_Code>5</IQ_Error_Code>
          <IQ_Error_Description>Record Count Zero. No Records To Retrieve
          </IQ_Error_Description>
          <IQ_Error_Extended_Data />
      </IQ_Error_Data_Item>
    </IQ_Error_Data>
  </IQ_API_Error>
</IQ_API_Result>
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":5,
         "IQ_Error_Description":"Record Count Zero. No Records To Retrieve",
         "IQ_Error_Data":{
            "IQ_Error_Data_Items":[
               {
                  "IQ_Error_Code":5,
                  "IQ_Error_Description":"Record Count Zero. No Records To Retrieve"
               }
            ]
         }
      }
   ]
}
```

Import Export Object: Creditor Journal

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Creditor_Journal>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Creditor_Journal>
</IQ_API>

<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
    <IQ_API_Error>
        <IQ_Error_Code>0</IQ_Error_Code>
    </IQ_API_Error>
    <IQ_API_Result_Data>
        <IQ_Root_XML>
            <IQ_Identification_Info>
                <Company_Store_ID/>
                <Company_Code>001</Company_Code>
                <Company_Name>test</Company_Name>
                <Company_Address1/>
                <Company_Address2/>
                <Company_Address3/>
                <Company_Address4/>
                <Company_Telephone1/>
                <Company_Telephone2/>
                <Company_Fax/>
                <Company_Email/>
                <Company_Tax/>
                <Company_Registration_Number/>
                <Company_Customs_Code/>
            </IQ_Identification_Info>
            <Creditor_Journal>
                <Creditor_Account>TEST</Creditor_Account>
                <Branch/>
                <Department/>
                <Date>2018-12-26</Date>
                <Reference>REference1</Reference>
                <Order_Number>OrderNumber1</Order_Number>
                <Notes>Notes1</Notes>
                <Journal_Code>IN</Journal_Code>
                <Ledger_Account>1350.000.000.00</Ledger_Account>
                <Amount>100</Amount>
                <Currency_Rate>1</Currency_Rate>
                <Vat_Rate>1</Vat_Rate>
                <Splits/>
            </Creditor_Journal>
        </IQ_Root_XML>
    </IQ_API_Result_Data>
</IQ_API_Result>

{
   "IQ_API":{
      "IQ_API_Request_Creditor_Journal":{
         "IQ_Company_Number":"001",
```

```
        "IQ_Terminal_Number":1,
        "IQ_User_Number":99,
        "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
        "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Creditor_Journals":[
            {
               "Branch":"",
               "Department":"",
               "Date":"2018-12-26",
               "Reference":"REference1",
               "Order_Number":"OrderNumber1",
               "Notes":"Notes1",
               "Journal_Code":"IN",
               "Ledger_Account":"1350.000.000.00",
               "Amount":100,
               "Currency_Rate":1,
               "Vat_Rate":1,
               "Splits":[

               ],
               "Creditor_Account":"TEST"
            }
         ]
      }
   }
}
```

Import Export Object: Creditors Master

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Creditor>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
<IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Creditor>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Creditor>
    <ID_Number/>
    <Date_Of_Birth>1899-12-30</Date_Of_Birth>
    <Title/>
    <Initials/>
    <Alternative_Name/>
    <Postal_Address_Details>
     <Address/>
     <Address/>
     <Address/>
     <Address/>
     <Postal_Code/>
    </Postal_Address_Details>
    <Delivery_Address_Details>
     <Address/>
     <Address/>
     <Address/>
     <Address/>
     <Postal_Code/>
    </Delivery_Address_Details>
    <Contact_Person/>
```

```
      <Credit_Limit>0</Credit_Limit>
      <Telephone_Numbers>
       <Telephone_Number/>
       <Telephone_Number/>
      </Telephone_Numbers>
      <Cellphone_Number/>
      <Email_Address/>
      <Allow_Use_Of_Email_Address>False</Allow_Use_Of_Email_Address>
      <Fax_Number/>
      <Tax_Number/>
      <Area/>
      <Total_Balance>0</Total_Balance>
      <Balance_Current>0</Balance_Current>
      <Balance_30_Days>0</Balance_30_Days>
      <Balance_60_Days>0</Balance_60_Days>
      <Balance_90_Days>0</Balance_90_Days>
      <Balance_120_Days>0</Balance_120_Days>
      <Balance_150_Days>0</Balance_150_Days>
      <Balance_180_Days>0</Balance_180_Days>
      <Status/>
      <Settlement_Discount_Percentage>0</Settlement_Discount_Percentage>
      <Company_Registration_Number/>
      <Bank_Name/>
      <Bank_Branch_Code/>
      <Bank_Branch_EFT_Number/>
      <Bank_Account_Number/>
      <Bank_Sub_Account_Number/>
      <Require_Bank_Proof_Of_Payment>False</Require_Bank_Proof_Of_Payment>
      <Bank_Account_Type/>
      <Group_Account/>
      <Currency>ZAR</Currency>
      <Comment/>
      <Cashier>1</Cashier>
      <Picture/>
      <Picture_Name/>
      <Is_Exclusive>False</Is_Exclusive>
      <Payment_Method/>
      <Allow_SMS_Marketing>False</Allow_SMS_Marketing>
      <LastPayment_Date>1899-12-30 00:00:00</LastPayment_Date>
      <LastPayment_Amount>0</LastPayment_Amount>
      <Created>2018-12-26 21:32:59</Created>
      <Modified>1899-12-30 00:00:00</Modified>
      <Creditor_Account>TEST</Creditor_Account>
      <Creditor_Name/>
      <Creditor_Group/>
      <Creditor_Sub_Group/>
      <Creditor_Sub_Group_Description/>
      <Terms/>
      <Vat_Status>Registered Vendor</Vat_Status>
      <GRV_Layout>1</GRV_Layout>
      <Special_Price_List_Number>0</Special_Price_List_Number>
     </Creditor>
    </IQ_Root_XML>
   </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Creditor":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

Import Export Object: Creditor Store Departments

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Creditor_Store_Department>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Creditor_Store_Department>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Creditor_Store_Department>
        <Department>CR_DEPT1</Department>
        <Description>CR_DEPT1_Desc</Description>
        <Created_Date>2018-12-28 07:30:30</Created_Date>
        <Modified_Date>1899-12-30 00:00:00</Modified_Date>
      </Creditor_Store_Department>
      <Creditor_Store_Department>
        <Department>CR_DEPT2</Department>
        <Description>CR_DEPT2_Desc</Description>
        <Created_Date>2018-12-28 07:30:35</Created_Date>
        <Modified_Date>1899-12-30 00:00:00</Modified_Date>
      </Creditor_Store_Department>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Creditor_Store_Department":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
```

```
        "IQ_User_Number":99,
        "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
        "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Creditor_Store_Departments":[
            {
               "Department":"CR_DEPT1",
               "Description":"CR_DEPT1_Desc",
               "Created_Date":"2018-12-28 07:30:30",
               "Modified_Date":"1899-12-30 00:00:00"
            },
            {
               "Department":"CR_DEPT2",
               "Description":"CR_DEPT2_Desc",
               "Created_Date":"2018-12-28 07:30:35",
               "Modified_Date":"1899-12-30 00:00:00"
            }
         ]
      }
   }
}
```

Import Export Object: Creditor Store Department Associations

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Creditor_Store_Department_Associations>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Creditor_Store_Department_Associations>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Creditor_Store_Department_Association>
        <Store_Department>CR_DEPT1</Store_Department>
        <Modified>2018-12-28 07:36:16</Modified>
        <Creditor_Account>TEST</Creditor_Account>
      </Creditor_Store_Department_Association>
      <Creditor_Store_Department_Association>
        <Store_Department>CR_DEPT2</Store_Department>
        <Modified>2018-12-28 07:36:24</Modified>
        <Creditor_Account>TEST2</Creditor_Account>
      </Creditor_Store_Department_Association>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Creditor_Store_Department_Associations":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
```

```
        "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
        "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Creditor_Store_Department_Associations":[
            {
               "Store_Department":"CR_DEPT1",
               "Modified":"2018-12-28 07:36:16",
               "Creditor_Account":"TEST"
            },
            {
               "Store_Department":"CR_DEPT2",
               "Modified":"2018-12-28 07:36:24",
               "Creditor_Account":"TEST2"
            }
         ]
      }
   }
}
```

Import Export Object: Creditor Price Lists

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Creditor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Creditor_Price_List>
</IQ_API>

<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Creditor_Stock_Price_List>
    <Price_List>
     <Price_List_Number>1</Price_List_Number>
     <Description>CRED_PRICE_LIST</Description>
     <Is_Inclusive>True</Is_Inclusive>
    </Price_List>
    <Items>
     <Item>
      <Stock_Code>123123</Stock_Code>
      <Prices>
       <Price>
        <Start_Date>1899-12-30</Start_Date>
        <End_Date>2019-01-05</End_Date>
        <Amount>100</Amount>
        <Discount>10</Discount>
       </Price>
       <Price>
        <Start_Date>1899-12-30</Start_Date>
        <End_Date>2019-01-13</End_Date>
        <Amount>200</Amount>
        <Discount>0</Discount>
```

```
        </Price>
       </Prices>
      </Item>
     </Items>
    </Creditor_Stock_Price_List>
   </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Creditor_Price_List":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Creditor_Stock_PriceLists":[
            {
               "Price_List":{
                  "Price_List_Number":1,
                  "Description":"CRED_PRICE_LIST",
                  "Is_Inclusive":true
               },
```

```
"Items":[
   {
      "Stock_Code":"123123",
      "Prices":[
         {
            "Start_Date":"1899-12-30",
            "End_Date":"2019-01-05",
            "Amount":100,
            "Discount":10
         },
         {
            "Start_Date":"1899-12-30",
            "End_Date":"2019-01-13",
            "Amount":200,
            "Discount":0
         }
      ]
   }
]
}
}
}
```

Import Export Object: Debtor Journal

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Debtor_Journal>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Debtor_Journal>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Debtor_Journal>
        <Branch/>
        <Department/>
        <Date>2018-12-28</Date>
        <Reference>123</Reference>
        <Order_Number>123</Order_Number>
        <Notes>123</Notes>
        <Journal_Code>IN</Journal_Code>
        <Ledger_Account/>
        <Amount>123</Amount>
        <Currency_Rate>1</Currency_Rate>
        <Vat_Rate>1</Vat_Rate>
        <Splits>
          <Split>
            <Ledger_Account>2000.000.000.00</Ledger_Account>
            <Reference>123</Reference>
            <Vat_Rate>1</Vat_Rate>
            <Amount>120</Amount>
            <Branch/>
            <Department/>
          </Split>
          <Split>
            <Ledger_Account>2001.000.000.00</Ledger_Account>
```

```
            <Reference>123a</Reference>
            <Vat_Rate>1</Vat_Rate>
            <Amount>3</Amount>
            <Branch/>
            <Department/>
          </Split>
        </Splits>
        <Debtor_Account>ASDASD</Debtor_Account>
        <Sales_Representative_Number>1</Sales_Representative_Number>
      </Debtor_Journal>
      <Debtor_Journal>
        <Branch/>
        <Department/>
        <Date>2018-12-28</Date>
        <Reference>456</Reference>
        <Order_Number>456</Order_Number>
        <Notes>456</Notes>
        <Journal_Code>IN</Journal_Code>
        <Ledger_Account>2000.000.000.00</Ledger_Account>
        <Amount>456</Amount>
        <Currency_Rate>1</Currency_Rate>
        <Vat_Rate>1</Vat_Rate>
        <Splits/>
        <Debtor_Account>ASDASD</Debtor_Account>
        <Sales_Representative_Number>1</Sales_Representative_Number>
      </Debtor_Journal>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Debtor_Journal":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
```

```
    "Company_Address3":"",
    "Company_Address4":"",
    "Company_Telephone1":"",
    "Company_Telephone2":"",
    "Company_Fax":"",
    "Company_Email":"",
    "Company_Tax":"",
    "Company_Registration_Number":"",
    "Company_Customs_Code":""
},
"Debtor_Journals":[
   {
      "Branch":"",
      "Department":"",
      "Date":"2018-12-28",
      "Reference":"123",
      "Order_Number":"123",
      "Notes":"123",
      "Journal_Code":"IN",
      "Ledger_Account":"",
      "Amount":123,
      "Currency_Rate":1,
      "Vat_Rate":1,
      "Splits":[
         {
            "Ledger_Account":"2000.000.000.00",
            "Reference":"123",
            "Vat_Rate":1,
            "Amount":120,
            "Branch":"",
            "Department":""
         },
         {
            "Ledger_Account":"2001.000.000.00",
            "Reference":"123a",
            "Vat_Rate":1,
            "Amount":3,
            "Branch":"",
            "Department":""
         }
      ],
      "Debtor_Account":"ASDASD",
      "Sales_Representative_Number":1
   },
   {
      "Branch":"",
      "Department":"",
```

```
            "Date":"2018-12-28",
            "Reference":"456",
            "Order_Number":"456",
            "Notes":"456",
            "Journal_Code":"IN",
            "Ledger_Account":"2000.000.000.00",
            "Amount":456,
            "Currency_Rate":1,
            "Vat_Rate":1,
            "Splits":[

            ],
            "Debtor_Account":"ASDASD",
            "Sales_Representative_Number":1
        }
    ]
  }
 }
}
```

Import Export Object: Debtors Master

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Debtor>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Debtor>
    <ID_Number>830</ID_Number>
    <Date_Of_Birth>1899-12-30</Date_Of_Birth>
    <Title>titl</Title>
    <Initials>ini</Initials>
    <Alternative_Name>addname</Alternative_Name>
    <Postal_Address_Details>
     <Address>po</Address>
     <Address>box</Address>
     <Address/>
     <Address/>
     <Postal_Code/>
    </Postal_Address_Details>
    <Delivery_Address_Details>
     <Address>del1</Address>
     <Address>2</Address>
     <Address>3</Address>
     <Address>4</Address>
     <Postal_Code/>
    </Delivery_Address_Details>
    <Additional_Addresses>
```

```
 <Additional_Address>
  <Branch_Number>12</Branch_Number>
  <Company_Name>123</Company_Name>
  <Contact_Name>12</Contact_Name>
  <Contact_Number>12</Contact_Number>
  <Address1>312</Address1>
  <Address2>321</Address2>
  <Address3>321</Address3>
  <Address4>321</Address4>
  <Postal_Code/>
  <Cellphone>213</Cellphone>
  <Fax_Number>21321</Fax_Number>
  <Email_Address>312</Email_Address>
  <Email_Modules>
   <Module_Name>Invoices and/or Recurring Charges</Module_Name>
   <Module_Name>Credit Notes</Module_Name>
   <Module_Name>Sales Orders</Module_Name>
   <Module_Name>Quotes</Module_Name>
   <Module_Name>Job Cards</Module_Name>
   <Module_Name>Debtors Credit Control</Module_Name>
   <Module_Name>Debtor Enquiries</Module_Name>
   <Module_Name>Email Marketing</Module_Name>
   <Module_Name>Statements</Module_Name>
   <Module_Name>Recall Documents</Module_Name>
  </Email_Modules>
 </Additional_Address>
</Additional_Addresses>
<Contact_Person>021</Contact_Person>
<Credit_Limit>1000</Credit_Limit>
<Telephone_Numbers>
 <Telephone_Number>021</Telephone_Number>
 <Telephone_Number>021</Telephone_Number>
</Telephone_Numbers>
<Cellphone_Number/>
<Email_Address>danie@simplydot.co.za</Email_Address>
<Allow_Use_Of_Email_Address>True</Allow_Use_Of_Email_Address>
<Fax_Number>021f</Fax_Number>
<Tax_Number/>
<Area>AREA</Area>
<Total_Balance>601</Total_Balance>
<Balance_Current>579</Balance_Current>
<Balance_30_Days>22</Balance_30_Days>
<Balance_60_Days>0</Balance_60_Days>
<Balance_90_Days>0</Balance_90_Days>
<Balance_120_Days>0</Balance_120_Days>
<Balance_150_Days>0</Balance_150_Days>
<Balance_180_Days>0</Balance_180_Days>
<Status>S</Status>
<Settlement_Discount_Percentage>0</Settlement_Discount_Percentage>
<Company_Registration_Number>reg</Company_Registration_Number>
<Bank_Name>b</Bank_Name>
<Bank_Branch_Code>bbc</Bank_Branch_Code>
<Bank_Branch_EFT_Number>ben</Bank_Branch_EFT_Number>
<Bank_Account_Number>ba</Bank_Account_Number>
<Bank_Sub_Account_Number>bsa</Bank_Sub_Account_Number>
<Require_Bank_Proof_Of_Payment>False</Require_Bank_Proof_Of_Payment>
```

<Bank_Account_Type>Current</Bank_Account_Type>
<Group_Account/>
<Currency>ZAR</Currency>
<Comment/>
<Cashier>1</Cashier>

<Picture>/9j/4AAQSkZJRgABAQEBLAEsAAD/4QAWRXhpZgAATU0AKgAAAAgAAAAAAAD/2wBDAAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkME
Q8SEhEPERETFhwXExQaFRERGCEYGh0dHx8fExciJCIeJBweHx7/2wBDAQUFBQcGBw4ICA4eFBEUHh4eHh4eHh4eHh4eHh4eHh4eHh4eHh4e
Hh4eHh4eHh4eHh4eHh4eHh4eHh7/wAARCABJAJ8DASIAAhEBAxEB/8QAHAAABAAIDAQEBAAAAAAAAAAAAAAYHAgUIBAMJ/8QAQxAAAQMDAgMEB
wQGCAcAAAAAAQIDBAAFEQYHEiExE0FRYQgUIjJxgZEVFlKxIzNidKHSJDVWcpOUsuE0NkJUc9Hw/8QAGwEAAgIDAQAAAAAAAAAAAAAAAAYEBwIDBQH
/xAAxEQABAgUDAQcCBwEBAAAAAAABAhEAAwQFIRIxQQYTFFFhcYGRB7EiMmKh0eHwQrL/2gAMAwEAAhEDEQA/AOy6UpRBClKUQQrV6tv9t0vp2bfr
s92UOI2VrPeo9AlPionAA8TW0rjf0odzPvdqL7u2iRxWS2OEFaD7Ml8civzSnmE/M94qdb6JVXOCOBv6RCr6xNJKKueIrncXVty1tq2ZqC5qwt5WGmgcpZa
HuoT5AfUknvqO0pVgIQlCQlIYCERa1LUVKOTClK+0GM9NmsQ46ON59xLTafFSjgD6mha0oSVKLAQJSVFhvHxpV8WXZywsQki6yZUuUR7ZbXwISf2RjPzJ+l
QjdTb1GlWGrlbpLj8Bxzs1JdxxtKIJHMdQcHuGKRrX9R7FdK8UMiYdRLJJBAUfI/ZwHhirOlLlR0xqZiQw3YuR6/08V9SlKe4W4UpSiCFKUoghX1jfr0/P8q+Vf
WN+vT8/yrw7R6N4/SulKVWEWTClKjW5esLdobSEu/3EhXZjgjs5wp90+6gfHqT3AE91ZIQqYoJSHJjFa0oSVKLARW/pTbmfdewfdazyOG83No9qtB9qNHPI
nyUrmB4DJ5cq48rZaovlx1Jf5l8uz5emTHS44ruHgkDuAGAB3ACtbVgW+iTSSQgb8+sIlfWKqppWduIyabcdWG2kKWs9EpGSa+/2fP8A+yk/4Sv/AFV3bHaP
+zLcNQ3BrEyWj+jpUObTR7/ir8seJqzqqXqH6vyrbcJlJSyBNSjBVqZzy34TgbPz6Q5WvoZdXTJnzpmgqyzPjh8iOPkMvLdLSGnFODqkJJI+VeuA3c4E5idHiyEP
R3EutqLR5KScg/UV0bojSbNil3O5vBK59wkuuKWOfZtlZKUD+BPn8Kkc7/gX/wDxq/Koly+sklNQaeRTCYjAcqZ3GcaTh8ee8b6ToOYZQmzJ2lW7M7eHO/2i
DWHdKzzLe27cIVwiPlPtJTGU4hR8UqSOnxxUC3e12NQss2mDDkRoaHO1UuQjhW6oAgYHckZPx8sVbe2v/INk/dEflWi1pon7166hvzVLRbIsJPa8JwXVFa8I
B7vEn4eOaUrDcenLV1DNmzKcoTJK2JXqAKXA0p0gknYOosfR47dypbrWWtCETQorCXGlnByXLnA5YCOem21uK4G0KWo9yRk0cbcaVwOIUhXgoYNdWtt
af0tbPZTBtUNOBxEpbBPmT1P1NeeRcdKXyzyZD8m2XCCwgqfKilwNjHUjqKckfWJcxeuXb1mU7atWf/LP5aveOAroRKE6VVKQtnZv7dvNo5YrJKFrzwpUrH
XAzV5aA0Bp+Y85qR+3q9TkuFdvhPniShruUoHqT1AOQAR1qey7rYLKURJVxttvJHsNLdQ3y8hkcq6N2+rUimqu60NKqcsfmyzHkYCnI2PD7EjMRqHomZNk
9tUTghJ2w7jg5lZ9xz6RyozFlPI42Yzzic4yhBIz8qxfYfYIDzLjRPQLSRn6117HUwtoOxy2ptz2wtsghWe/I6/GqW9JD+t7R+7uf6hXvSv1QXf7si3ml0BWrOpyG
BO2keEeXno9NtolVQnambDNuW8TFT1mySlwEVhWTfvircMJEfpdSlKq+LJjF51tllbzziW220lS1qOAkDmST3CuHPSD3Hc1/q9QhuKFjt5U1BR0Dn4nSPFW
OXgAO/NWz6W25nqcVWgbLIxIfSFXRxB5obPMM/FXIq8sDvNct02WK36E94WMnb08feFa9V+s9gg4G/8AEKk2g9OOXqaZDpLcSOQSrhB4ldQnBBB8/wD
eo4ylCnUJcXwIKgFKxnA7zjvqzrVq/S1st7UKL6ylppOB+i5k95Pma09WV1wp6TsrfKUqYvDgOEjk+p4+Yl9JUFuqKvtbjNSmWjLEsVHgeg5+ImomX8AAagkgf
u7H8lPXb/8A2hk/5dj+Sq6uGuI0jUUBTReRbY5K3Dw+0tRSQOXgM/8A3Kt19/tP/jk/4X+9UzUdJXSUlB7oCVB8SkFskMfw74f3i3Ke99OzlTAZwSElgStQfALj
OzlvaJX67f8A+0Mn/LsfyV85Uy/GM6DqCSRwHI9XY58v7lV6xriO7qpUqSXW7c2yptlCU5JUSPaI8Tj5VuH9eWBbK0BcnKkkD9F5fGsp3SVzp1oHdApwCWll
LPw+nccxhIvfT1QhZ7YJYkB1qDtyz7HiNxoyXekaVtqGL0+y0I6QlAYaISPDJQT9a2guV7LymRqJ/tEpCin1djOCSAfc8jUG07rSyQbHDhvqkdqy0Eq4W8jIrTan
1aHNQxbpZHXElpjs1hxGAr2icEd4ronpCvr7jOSqnCQSshSpaWJckOdPPj7xzzfrJQ26SvXrUAgKSFl9gCw1ceHtG+3PtWorz6rJEt65pYSpPZlKElGe8BIAOfryq
FK05qGPbZEtUJ9lgAB1JOFKTnPNPUgEA/x7qmlu3FtzjQE6JIYc7y2AtP5g1lP3EtTbZESLJkLxyCgEJ+vM/wAKYbTV9TWuSigRRApSfAAM77ghI9feONcrf0pcJ
i63vhTqGzuXZtiCr2+0SbTl0u8iwwXYl+eSyWEBCUx2clwMcPud2MfKq41dpnUcnUEuYpp24esOFYfBGTnuI5Yx06Y8K81g1hLtU19YjNKhvuFaoyPZSgn8Hh
8Kl7O4ViW3xONzG1d6S2D+RqPLtd76cr5lRQ0yVpX4JS7Eux0sXHwd/TYKrpy/UKJNVOMpaN8nLYcanDHdtx99noONqGy6ebhLuzsX21LDKUNrDee7Kkn4
8jjnUT3jenPXC3mbOclqDSuErbQnhGR+ECt21uDYVglaZbeDgAtg5HjyNQ/cK+Qb3LiuwS4UtNqSrjTjmTR0zbbmeoRW1lNo1aiVaAMkH/pn/fPMHUs+xosJ
p6GcFKGkAaiSWI4dv29Ii9ZN++KxrJv3xVyGKgj9Lqg29ev4u32jXrkShy5SMs29hX/W5j3iPwp6n5DvFS+7XCHabZJudxkIjxIrSnXnVnkhIGSa4N3j15M3A1k/
d3eNqE3lqBHJ/VNA8s/tK6nzOOgFIlqoO9zXV+Ub/wAQ7XSu7rKZP5jt/MRO5TZVxuEi4Tn1yJUlxTrzqzlS1qOST8689KU9ANgQkkvkxLNn7fCuu5+nrdcYzc
mJInIQ804MpWk9Qa6/1DtnoG22aRNg7cwLpIaAKIjKEpW7kgYBUcDAOefhXHu0lzg2XcuwXW5yBHhRZqHHnSkkISOpwAT9K6x1Pu1tde7FKtaNwJFsU+k
ASoLb7b7eFA5Srg5dMfAmly8pnmejs3Zss/j5QwWkyBIX2jO+Hbw84ym7TbYXDTAn3nSEfT2WO0ezJ7JUXlz4lJXwcvPIqhPR801pm871XGyyWGb3ZmWZ
Xqyn0cnUpcSEOY8Snn86u6zbw7XaY063AOubpf1tBR7WUy69Idyc4KihI8hnFUJZ90bZY99JeurRYvV7RKUptcFHClfZqSkKUMeyFlSePHTJlz31rokVapc5BC
sjDuPh8xsrF0qZkpY04OWY/LYi2fSY0Do3T21ki5WTTkCBMTKZQHmW8KAKuYr3ejzt5oi+7P2S63fTNvmzX/WO1fdbypXDIcSMnyAA+VbC+7nbL7gaYXZ79
e1NRnila2H2nWXEKScg8SRjPwJFZW7dnZ3QelY9lsV3XJiw0q7CNFZddWriUVH2lgJyVKJ5qHWohVVmmEnSvXqd87N4xKCaUVJnak6NLNjd/CKg1tpjT8P0
pbdpyLaYrVoclw0LiJT+jUFISVDHnk1de5m2OgLftzqSfC0nbGJUa1SXWXUNYUhaWlFKh5giucxr6NqD0gYGuLqlNugi4sOKByvsWW+FIzgZJwnJwOpNdAbj
7wbb3Tb3UVsganZfly7XJYYbEd4Fa1NKCRkoxzJHWpdZLqkqkAPgB2ffziLSTKVSZxLZJZ228ogHolaM0tqfTd7kagscO5OszEIbU+jiKUlGcCtL6XGltO6Xuun29
P2iLbUSGHlOhhHCFkKRgn6mtj6Kmv8ASGj9O3mLqS9N296RLQ40lTTiuJIRgn2Unvqea71X6P2uH4r2pr01NXESpLJAlt8IUQT7gGeg61kuZPk3EzClRR5Anj4
jxEuRNoAgKSF+ZA5+Yon0arJadQbqxLbeoDE6GuM8pTLycpJCCQa6V1vozaLR+m5F/vOjremFHUhLhajcavaUEjAz4kVTGlb5thpL0go1007cERdMIt60l4h9
wB1SCCPbBX1x5Vbestytl9W6ffsV71Q27BfKFOIQ1IQTwqChzCM9QK13BU+bUIWkL0EB2d98+8Z0CZMunWhRTqBLO3t7R4bLtpspuRp1V103biy0VlovR
HXWXGlgAlJQrKc4IPNJ61zhvRt/I261f9jrleuRH2hIiPlPCVNkkYUO5QIIPyPfiuibRuzspt/p42zSrzzzIUXDHiRnVLcWQAVKW7gE8gMlXQCuc94tfzNxdWm8yI
yYkdloMRI4Vxdm2CTzPeokkk/Ad1SbWKvty+rs/wBW/l/tojXI0vYhtPafp28/9vELrJs4WM1jSmGODHQvpabmfadwVoSyyMwoiwbk4g8nXh0a+CTzP7X92
ueq9t//AK9uH7y5/qNeKotHTIppIloiTV1C6iaVqhSlKlRGhSlKIIUpSiCFKUoghSlKIIUpSiCFKUoghSlKIIUpSiCP/9m=</Picture>
<Picture_Name/>
<Is_Exclusive>False</Is_Exclusive>
<Payment_Method/>
<Allow_SMS_Marketing>False</Allow_SMS_Marketing>
<LastPayment_Date>1899-12-30 00:00:00</LastPayment_Date>
<LastPayment_Amount>0</LastPayment_Amount>
<Created>2018-10-20 10:56:28</Created>

```
    <Modified>2018-12-28 07:20:47</Modified>
    <Store_Departments>
     <Store_Department>
      <Name>DR_DEPT1</Name>
      <Description>DR_DEPT1_Desc</Description>
      <Modified_Date>2018-12-28 07:20:47</Modified_Date>
     </Store_Department>
    </Store_Departments>
    <Debtor_Account>ASDASD</Debtor_Account>
    <Debtor_Name>nmae</Debtor_Name>
    <Debtor_Group/>
    <Debtor_Sub_Group/>
    <Debtor_Sub_Group_Description/>
    <Terms/>
    <Delivery_Route>001</Delivery_Route>
    <Credit_Limit_Insured>0</Credit_Limit_Insured>
    <Credit_Limit_Reserved>0</Credit_Limit_Reserved>
    <Preferred_Sell_Price>Retail Price</Preferred_Sell_Price>
    <Invoice_Discount_Percentage>10</Invoice_Discount_Percentage>
    <Apply_As_Line_Discount>False</Apply_As_Line_Discount>
    <Normal_Representative>1</Normal_Representative>
    <Sales_Manager/>
    <Invoice_Layout>1</Invoice_Layout>
    <Vat_Status>Normal</Vat_Status>
    <Charge_Interest>True</Charge_Interest>
    <Interest_Rate>0</Interest_Rate>
    <Interest_Risk_Profile>0</Interest_Risk_Profile>
    <Export_Status/>
    <Allow_Cash_Sale>False</Allow_Cash_Sale>
    <Require_Invoice_Order_Number>False</Require_Invoice_Order_Number>
    <Notification/>
    <Own_Price_List_Number>0</Own_Price_List_Number>
    <Loyalty_Account/>
   </Debtor>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Debtor":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
```

```
"IQ_API_Result_Data":{
   "IQ_Root_JSON":{
      "IQ_Identification_Info":{
         "Company_Store_ID":"",
         "Company_Code":"001",
         "Company_Name":"test",
         "Company_Address1":"",
         "Company_Address2":"",
         "Company_Address3":"",
         "Company_Address4":"",
         "Company_Telephone1":"",
         "Company_Telephone2":"",
         "Company_Fax":"",
         "Company_Email":"",
         "Company_Tax":"",
         "Company_Registration_Number":"",
         "Company_Customs_Code":""
      },
      "Debtors_Master":[
         {
            "ID_Number":"830",
            "Date_Of_Birth":"1899-12-30",
            "Title":"titl",
            "Initials":"ini",
            "Alternative_Name":"addname",
            "Postal_Address_Details":[
               "po",
               "box",
               "",
               "",
               ""
            ],
            "Delivery_Address_Details":[
               "del1",
               "2",
               "3",
               "4",
               ""
            ],
            "Additional_Addresses":[
               {
                  "Branch_Number":"12",
                  "Company_Name":"123",
                  "Contact_Name":"12",
                  "Contact_Number":"12",
                  "Address1":"312",
                  "Address2":"321",
```

```
      "Address3":"321",
      "Address4":"321",
      "Postal_Code":"",
      "Cellphone":"213",
      "Fax_Number":"21321",
      "Email_Address":"312",
      "Email_Modules":[
         "Invoices and\/or Recurring Charges",
         "Credit Notes",
         "Sales Orders",
         "Quotes",
         "Job Cards",
         "Debtors Credit Control",
         "Debtor Enquiries",
         "Email Marketing",
         "Statements",
         "Recall Documents"
      ]
   }
],
"Contact_Person":"021",
"Credit_Limit":1000,
"Telephone_Numbers":[
   "021",
   "021"
],
"Cellphone_Number":"",
"Email_Address":"danie@simplydot.co.za",
"Allow_Use_Of_Email_Address":true,
"Fax_Number":"021f",
"Tax_Number":"",
"Area":"AREA",
"Total_Balance":601,
"Balance_Current":579,
"Balance_30_Days":22,
"Balance_60_Days":0,
"Balance_90_Days":0,
"Balance_120_Days":0,
"Balance_150_Days":0,
"Balance_180_Days":0,
"Status":"S",
"Settlement_Discount_Percentage":0,
"Company_Registration_Number":"reg",
"Bank_Name":"b",
"Bank_Branch_Code":"bbc",
"Bank_Branch_EFT_Number":"ben",
```

```
"Bank_Account_Number":"ba",
"Bank_Sub_Account_Number":"bsa",
"Require_Bank_Proof_Of_Payment":false,
"Bank_Account_Type":"Current",
"Group_Account":"",
"Currency":"ZAR",
"Comment":"",
"Cashier":1,
"Picture":"",
"Picture_Name":"",
"Is_Exclusive":false,
"Payment_Method":"",
"Allow_SMS_Marketing":false,
"LastPayment_Date":"1899-12-30 00:00:00",
"LastPayment_Amount":0,
"Created":"2018-10-20 10:56:28",
"Modified":"2018-12-28 07:20:47",
"Store_Departments":[
   {
      "Name":"DR_DEPT1",
      "Description":"DR_DEPT1_Desc",
      "Modified_Date":"2018-12-28 07:20:47"
   }
],
"Debtor_Account":"ASDASD",
"Debtor_Name":"nmae",
"Debtor_Group":"",
"Debtor_Sub_Group":"",
"Debtor_Sub_Group_Description":"",
"Terms":"",
"Delivery_Route":"001",
"Credit_Limit_Insured":0,
"Credit_Limit_Reserved":0,
"Preferred_Sell_Price":"Retail Price",
"Invoice_Discount_Percentage":10,
"Apply_As_Line_Discount":false,
"Normal_Representative":1,
"Sales_Manager":"",
"Invoice_Layout":1,
"Vat_Status":"Normal",
"Charge_Interest":true,
"Interest_Rate":0,
"Interest_Risk_Profile":0,
"Export_Status":"",
"Allow_Cash_Sale":false,
```

```
                "Require_Invoice_Order_Number":false,
                "Notification":"",
                "Own_Price_List_Number":0,
                "Loyalty_Account":""
            }
        ]
    }
  }
}
```

Import Export Object: Debtor Store Departments

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor_Store_Department>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Debtor_Store_Department>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Debtor_Store_Department>
        <Department>DR_DEPT1</Department>
        <Description>DR_DEPT1_Desc</Description>
        <Created_Date>2018-12-28 07:18:58</Created_Date>
        <Modified_Date>1899-12-30 00:00:00</Modified_Date>
      </Debtor_Store_Department>
      <Debtor_Store_Department>
        <Department>DR_DEPT2</Department>
        <Description>DR_DEPT2_Desc</Description>
        <Created_Date>2018-12-28 07:19:05</Created_Date>
        <Modified_Date>1899-12-30 00:00:00</Modified_Date>
      </Debtor_Store_Department>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Debtor_Store_Department":{
         "IQ_Company_Number":"001",
```

```
        "IQ_Terminal_Number":1,
        "IQ_User_Number":99,
        "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
        "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Debtor_Store_Departments":[
            {
               "Department":"DR_DEPT1",
               "Description":"DR_DEPT1_Desc",
               "Created_Date":"2018-12-28 07:18:58",
               "Modified_Date":"1899-12-30 00:00:00"
            },
            {
               "Department":"DR_DEPT2",
               "Description":"DR_DEPT2_Desc",
               "Created_Date":"2018-12-28 07:19:05",
               "Modified_Date":"1899-12-30 00:00:00"
            }
         ]
      }
   }
}
```

Import Export Object: Debtor Store Department Associations

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Debtor_Store_Department_Associations>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Debtor_Store_Department_Associations>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Debtor_Store_Department_Association>
    <Store_Department>DR_DEPT1</Store_Department>
    <Modified>2018-12-28 07:20:47</Modified>
    <Debtor_Account>ASDASD</Debtor_Account>
   </Debtor_Store_Department_Association>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Debtor_Store_Department_Associations":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Debtor_Store_Department_Associations":[
            {
               "Store_Department":"DR_DEPT1",
               "Modified":"2018-12-28 07:20:47",
               "Debtor_Account":"ASDASD"
            }
         ]
      }
   }
}
```

Import Export Object: Debtor Price Lists

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Debtor_Price_List>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Debtor_Price_List>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Debtor_Stock_Price_List>
        <Price_List>
          <Price_List_Number>2</Price_List_Number>
          <Description>DR_PRCLIST</Description>
          <Is_Inclusive>False</Is_Inclusive>
        </Price_List>
        <Items>
          <Item>
            <Stock_Code>123123</Stock_Code>
            <Price_Type>0</Price_Type>
            <Prices>
              <Price>
                <Start_Date>1899-12-30</Start_Date>
                <End_Date>2018-12-31</End_Date>
                <Amount>50</Amount>
                <Discount>10</Discount>
              </Price>
              <Price>
                <Start_Date>2019-01-05</Start_Date>
                <End_Date>2019-02-06</End_Date>
                <Amount>40</Amount>
```

```
                <Discount>1</Discount>
              </Price>
            </Prices>
          </Item>
        </Items>
      </Debtor_Stock_Price_List>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Debtor_Price_List":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Debtor_Stock_PriceLists":[
            {
               "Price_List":{
                  "Price_List_Number":2,
                  "Description":"DR_PRCLIST",
                  "Is_Inclusive":false
```

```
            },
            "Items":[
               {
                  "Stock_Code":"123123",
                  "Prices":[
                     {
                        "Start_Date":"1899-12-30",
                        "End_Date":"2018-12-31",
                        "Amount":50,
                        "Discount":10
                     },
                     {
                        "Start_Date":"2019-01-05",
                        "End_Date":"2019-02-06",
                        "Amount":40,
                        "Discount":1
                     }
                  ]
               }
            ]
         }
      ]
   }
}
```

Import Export Object: Ledger Journal

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Ledger_Journal>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Ledger_Journal>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Ledger_Journal>
        <Date>2018-12-28</Date>
        <Account>1200.000.000.00</Account>
        <Reference>Advert Ref</Reference>
        <Description>Advertising</Description>
        <Notes>Line 1</Notes>
        <Debit>100</Debit>
        <Credit>0</Credit>
        <VatRate>1</VatRate>
      </Ledger_Journal>
      <Ledger_Journal>
        <Date>2018-12-28</Date>
        <Account>1350.000.000.00</Account>
        <Reference>Gen Exp</Reference>
        <Description>General Expenses</Description>
        <Notes>Line 2</Notes>
        <Debit>100</Debit>
        <Credit>0</Credit>
        <VatRate>1</VatRate>
      </Ledger_Journal>
      <Ledger_Journal>
        <Date>2018-12-28</Date>
        <Account>3700.000.000.00</Account>
```

```
        <Reference>out acc</Reference>
        <Description>Cash on Hand</Description>
        <Notes>Line 3</Notes>
        <Debit>0</Debit>
        <Credit>200</Credit>
        <VatRate>0</VatRate>
      </Ledger_Journal>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Ledger_Journal":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Ledger_Journals":[
            {
               "Date":"2018-12-28",
               "Account":"1200.000.000.00",
               "Reference":"Advert Ref",
```

```
            "Description":"Advertising",
            "Notes":"Line 1",
            "Debit":100,
            "Credit":0,
            "VatRate":1
         },
         {
            "Date":"2018-12-28",
            "Account":"1350.000.000.00",
            "Reference":"Gen Exp",
            "Description":"General Expenses",
            "Notes":"Line 2",
            "Debit":100,
            "Credit":0,
            "VatRate":1
         },
         {
            "Date":"2018-12-28",
            "Account":"3700.000.000.00",
            "Reference":"out acc",
            "Description":"Cash on Hand",
            "Notes":"Line 3",
            "Debit":0,
            "Credit":200,
            "VatRate":0
         }
      ]
   }
}
}
```

Import Export Object: Sales Representatives

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Sales_Rep>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Sales_Rep>
</IQ_API>


<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Sales_Representative>
        <REP_Number>1</REP_Number>
        <REP_Name>Rep 1</REP_Name>
        <Comm_Type>Sales</Comm_Type>
        <Target_Value_1>0</Target_Value_1>
        <Target_Value_2>0</Target_Value_2>
        <Target_Value_3>0</Target_Value_3>
        <Target_Value_4>0</Target_Value_4>
       <Target_Value_5>0</Target_Value_5>
        <Commission_Value_1>0</Commission_Value_1>
        <Commission_Value_2>0</Commission_Value_2>
        <Commission_Value_3>0</Commission_Value_3>
        <Commission_Value_4>0</Commission_Value_4>
        <Commission_Value_5>0</Commission_Value_5>
        <Email/>
        <Email_Status>True</Email_Status>
        <Cellphone/>
        <On_Hold>0</On_Hold>
        <Is_Sales_Manager>False</Is_Sales_Manager>
        <Assigned_To_Area_Manager>
```

```
          <Rep>2</Rep>
        </Assigned_To_Area_Manager>
      </Sales_Representative>
      <Area_Sales_Manager>
        <REP_Number>2</REP_Number>
        <REP_Name>MANAGER REP</REP_Name>
        <Comm_Type>Sales</Comm_Type>
        <Target_Value_1>0</Target_Value_1>
        <Target_Value_2>0</Target_Value_2>
        <Target_Value_3>0</Target_Value_3>
        <Target_Value_4>0</Target_Value_4>
        <Target_Value_5>0</Target_Value_5>
        <Commission_Value_1>0</Commission_Value_1>
        <Commission_Value_2>0</Commission_Value_2>
        <Commission_Value_3>0</Commission_Value_3>
        <Commission_Value_4>0</Commission_Value_4>
        <Commission_Value_5>0</Commission_Value_5>
        <Email>MANAGER@REP</Email>
        <Email_Status>False</Email_Status>
        <Cellphone>07777777777777777</Cellphone>
        <On_Hold>4194368</On_Hold>
        <Is_Sales_Manager>True</Is_Sales_Manager>
        <Assigned_Reps>
          <Rep>1</Rep>
        </Assigned_Reps>
      </Area_Sales_Manager>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Sales_Rep":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
```

```
    "Company_Address2":"",
    "Company_Address3":"",
    "Company_Address4":"",
    "Company_Telephone1":"",
    "Company_Telephone2":"",
    "Company_Fax":"",
    "Company_Email":"",
    "Company_Tax":"",
    "Company_Registration_Number":"",
    "Company_Customs_Code":""
},
"Sales_Representatives":[
    {
        "REP_Number":1,
        "REP_Name":"Rep 1",
        "Comm_Type":"Sales",
        "Target_Value_1":0,
        "Target_Value_2":0,
        "Target_Value_3":0,
        "Target_Value_4":0,
        "Target_Value_5":0,
        "Commission_Value_1":0,
        "Commission_Value_2":0,
        "Commission_Value_3":0,
        "Commission_Value_4":0,
        "Commission_Value_5":0,
        "Email":"",
        "Email_Status":true,
        "Cellphone":"",
        "On_Hold":0,
        "Is_Sales_Manager":false,
        "Assigned_To_Area_Manager":[
            {
                "Rep":2
            }
        ]
    },
    {
        "REP_Number":2,
        "REP_Name":"MANAGER REP",
        "Comm_Type":"Sales",
        "Target_Value_1":0,
        "Target_Value_2":0,
        "Target_Value_3":0,
        "Target_Value_4":0,
        "Target_Value_5":0,
```

```
                "Commission_Value_1":0,
                "Commission_Value_2":0,
                "Commission_Value_3":0,
                "Commission_Value_4":0,
                "Commission_Value_5":0,
                "Email":"MANAGER@REP",
                "Email_Status":false,
                "Cellphone":"0777777777777777",
                "On_Hold":4194368,
                "Is_Sales_Manager":true,
                "Assigned_Reps":[
                   {
                      "Rep":1
                   }
                ]
             }
          ]
       }
    }
}
```

Import Export Object: Stock Contract Pricing

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Stock_ContractPricing>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Stock_ContractPricing>
</IQ_API>

<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
  <IQ_API_Error>
    <IQ_Error_Code>0</IQ_Error_Code>
  </IQ_API_Error>
  <IQ_API_Result_Data>
    <IQ_Root_XML>
      <IQ_Identification_Info>
        <Company_Store_ID/>
        <Company_Code>001</Company_Code>
        <Company_Name>test</Company_Name>
        <Company_Address1/>
        <Company_Address2/>
        <Company_Address3/>
        <Company_Address4/>
        <Company_Telephone1/>
        <Company_Telephone2/>
        <Company_Fax/>
        <Company_Email/>
        <Company_Tax/>
        <Company_Registration_Number/>
        <Company_Customs_Code/>
      </IQ_Identification_Info>
      <Stock_Contract_Price_Group_Department>
        <Debtor_Group>001</Debtor_Group>
        <Contract_Type>Discount per Group per Department</Contract_Type>
        <Major_Department>001</Major_Department>
        <Minor_Department>0001</Minor_Department>
        <Discount_Percentage>10</Discount_Percentage>
        <Start_Date>2018-12-28</Start_Date>
        <End_Date>2018-12-28</End_Date>
        <Created>2018-12-28 08:39:24</Created>
        <Modified>1899-12-30 00:00:00</Modified>
      </Stock_Contract_Price_Group_Department>
      <Stock_Contract_Price_Group_Product>
        <Debtor_Group>001</Debtor_Group>
        <Contract_Type>Discount per Group per Product</Contract_Type>
        <Product>123123</Product>
        <Price_Exclusive>100</Price_Exclusive>
        <Price_Inclusive>115</Price_Inclusive>
        <Discount_Percentage>0</Discount_Percentage>
        <Start_Date>2018-12-28</Start_Date>
        <End_Date>2018-12-28</End_Date>
        <Created>2018-12-28 08:39:10</Created>
```

```
        <Modified>1899-12-30 00:00:00</Modified>
      </Stock_Contract_Price_Group_Product>
      <Stock_Contract_Price_Debtor_Department>
        <Debtor_Group>ASDASD</Debtor_Group>
        <Contract_Type>Discount per Department per Debtor</Contract_Type>
        <Major_Department>001</Major_Department>
        <Minor_Department>0001</Minor_Department>
        <Discount_Percentage>10</Discount_Percentage>
        <Start_Date>2018-12-28</Start_Date>
        <End_Date>2018-12-28</End_Date>
        <Created>2018-12-28 08:38:55</Created>
        <Modified>1899-12-30 00:00:00</Modified>
      </Stock_Contract_Price_Debtor_Department>
      <Stock_Contract_Price_Debtor_Product>
        <Debtor_Group>ASDASD</Debtor_Group>
        <Contract_Type>Discount per Product per Debtor</Contract_Type>
        <Product>123123</Product>
        <Price_Exclusive>100</Price_Exclusive>
        <Price_Inclusive>115</Price_Inclusive>
        <Discount_Percentage>0</Discount_Percentage>
        <Start_Date>2018-12-28</Start_Date>
        <End_Date>2018-12-28</End_Date>
        <Created>2018-12-28 08:38:44</Created>
        <Modified>1899-12-30 00:00:00</Modified>
      </Stock_Contract_Price_Debtor_Product>
    </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Stock_ContractPricing":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
```

```
    "Company_Address2":"",
    "Company_Address3":"",
    "Company_Address4":"",
    "Company_Telephone1":"",
    "Company_Telephone2":"",
    "Company_Fax":"",
    "Company_Email":"",
    "Company_Tax":"",
    "Company_Registration_Number":"",
    "Company_Customs_Code":""
},
"Stock_Contract_Pricing":[
    {
        "Debtor_Group":"001",
        "Contract_Type":"Discount per Group per Department",
        "Major_Department":"001",
        "Minor_Department":"0001",
        "Discount_Percentage":10,
        "Start_Date":"2018-12-28",
        "End_Date":"2018-12-28",
        "Created":"2018-12-28 08:39:24",
        "Modified":"1899-12-30 00:00:00"
    },
    {
        "Debtor_Group":"001",
        "Contract_Type":"Discount per Group per Product",
        "Product":"123123",
        "Price_Exclusive":100,
        "Price_Inclusive":115,
        "Discount_Percentage":0,
        "Start_Date":"2018-12-28",
        "End_Date":"2018-12-28",
        "Created":"2018-12-28 08:39:10",
        "Modified":"1899-12-30 00:00:00"
    },
    {
        "Debtor_Group":"ASDASD",
        "Contract_Type":"Discount per Department per Debtor",
        "Major_Department":"001",
        "Minor_Department":"0001",
        "Discount_Percentage":10,
        "Start_Date":"2018-12-28",
        "End_Date":"2018-12-28",
        "Created":"2018-12-28 08:38:55",
        "Modified":"1899-12-30 00:00:00"
    },
```

```
            {
               "Debtor_Group":"ASDASD",
               "Contract_Type":"Discount per Product per Debtor",
               "Product":"123123",
               "Price_Exclusive":100,
               "Price_Inclusive":115,
               "Discount_Percentage":0,
               "Start_Date":"2018-12-28",
               "End_Date":"2018-12-28",
               "Created":"2018-12-28 08:38:44",
               "Modified":"1899-12-30 00:00:00"
            }
         ]
      }
   }
}
```

Import Export Object: Stock Contract Pricing Itemized

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Stock_ContractPricing_Itemized>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Stock_ContractPricing_Itemized>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Stock_Contract_Price_Group_Department/>
   <Stock_Contract_Price_Group_Product/>
   <Stock_Contract_Price_Debtor_Department>
    <Stock_Contract_Price_Itemized>
     <Contract_Type>Discount per Department per Debtor</Contract_Type>
     <Debtor_Account>ASDASD</Debtor_Account>
     <Product>123123</Product>
     <Price_Exclusive>0</Price_Exclusive>
     <Price_Inclusive>0</Price_Inclusive>
     <Start_Date>2018-12-28</Start_Date>
     <End_Date>2018-12-28</End_Date>
     <Created>2018-12-28 08:38:55</Created>
     <Modified>1899-12-30 00:00:00</Modified>
    </Stock_Contract_Price_Itemized>
    <Stock_Contract_Price_Itemized>
     <Contract_Type>Discount per Department per Debtor</Contract_Type>
     <Debtor_Account>ASDASD</Debtor_Account>
     <Product>2</Product>
     <Price_Exclusive>0</Price_Exclusive>
     <Price_Inclusive>0</Price_Inclusive>
     <Start_Date>2018-12-28</Start_Date>
```

```
      <End_Date>2018-12-28</End_Date>
      <Created>2018-12-28 08:38:55</Created>
      <Modified>1899-12-30 00:00:00</Modified>
     </Stock_Contract_Price_Itemized>
     <Stock_Contract_Price_Itemized>
      <Contract_Type>Discount per Department per Debtor</Contract_Type>
      <Debtor_Account>ASDASD</Debtor_Account>
      <Product>3</Product>
      <Price_Exclusive>0</Price_Exclusive>
      <Price_Inclusive>0</Price_Inclusive>
      <Start_Date>2018-12-28</Start_Date>
      <End_Date>2018-12-28</End_Date>
      <Created>2018-12-28 08:38:55</Created>
      <Modified>1899-12-30 00:00:00</Modified>
     </Stock_Contract_Price_Itemized>
    </Stock_Contract_Price_Debtor_Department>
    <Stock_Contract_Price_Debtor_Product/>
   </IQ_Root_XML>
  </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Stock_ContractPricing_Itemized":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
```

```
   "Company_Tax":"",
   "Company_Registration_Number":"",
   "Company_Customs_Code":""
},
"Stock_Contract_Pricing_Itemized":[
   {
      "Stock_Contract_Price_Itemized":[

      ]
   },
   {
      "Stock_Contract_Price_Itemized":[

      ]
   },
   {
      "Stock_Contract_Price_Itemized":[
         {
            "Contract_Type":"Discount per Department per Debtor",
            "Debtor_Account":"ASDASD",
            "Product":"123123",
            "Price_Exclusive":0,
            "Price_Inclusive":0,
            "Start_Date":"2018-12-28",
            "End_Date":"2018-12-28",
            "Created":"2018-12-28 08:38:55",
            "Modified":"1899-12-30 00:00:00"
         },
         {
            "Contract_Type":"Discount per Department per Debtor",
            "Debtor_Account":"ASDASD",
            "Product":"2",
            "Price_Exclusive":0,
            "Price_Inclusive":0,
            "Start_Date":"2018-12-28",
            "End_Date":"2018-12-28",
            "Created":"2018-12-28 08:38:55",
            "Modified":"1899-12-30 00:00:00"
         },
         {
            "Contract_Type":"Discount per Department per Debtor",
            "Debtor_Account":"ASDASD",
            "Product":"3",
            "Price_Exclusive":0,
            "Price_Inclusive":0,
            "Start_Date":"2018-12-28",
            "End_Date":"2018-12-28",
            "Created":"2018-12-28 08:38:55",
            "Modified":"1899-12-30 00:00:00"
```

```
                    }
                ]
            },
            {
                "Stock_Contract_Price_Itemized":[

                ]
            }
        ]
    }
}
}
```

Import Export Object: Stock Master

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Stock>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>|
</IQ_API_Request_Stock>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Stock>
    <Stock_Code>123123</Stock_Code>
    <Barcode>123123</Barcode>
    <Multiple_Barcodes/>
    <General_Code/>
    <Description/>
    <Alternative_Description/>
    <Major_Department>001</Major_Department>
    <Minor_Department>0001</Minor_Department>
    <Category/>
    <Range/>
    <Item_Category>Stock Item</Item_Category>
    <Use_Fixed_Cost>False</Use_Fixed_Cost>
    <Cost_As_Percentage_Of_Sell_Price>0</Cost_As_Percentage_Of_Sell_Price>
    <Pack_Size>0</Pack_Size>
```

<Pack_Description/>
<Bin_Location/>
<DATE_LastMoved>2018-10-20</DATE_LastMoved>
<DATE_LastPurchased>1899-12-30</DATE_LastPurchased>
<DATE_LastSold>2018-10-20</DATE_LastSold>
<Date_LastTransferred>1899-12-30</Date_LastTransferred>
<Previous_SellingPrice>0</Previous_SellingPrice>
<Previous_SellingPrice_Date>2018-10-20</Previous_SellingPrice_Date>
<Is_Sell_Price_Inclusive>True</Is_Sell_Price_Inclusive>
<Sell_Prices>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>
<Sell_Price>
<Inclusive>0</Inclusive>
<Exclusive>0</Exclusive>
</Sell_Price>

```
</Sell_Prices>
<LayBye_Onhand>0</LayBye_Onhand>
<Average_Cost>0</Average_Cost>
<Latest_Cost>0</Latest_Cost>
<Base_Cost>0</Base_Cost>
<Previous_Cost>0</Previous_Cost>
<Highest_Cost>0</Highest_Cost>
<Minimum_Level>0</Minimum_Level>
<Maximum_Level>0</Maximum_Level>
<ReOrder_Level>0</ReOrder_Level>
<ReOrder_Quantity>0</ReOrder_Quantity>
<Store_Serial_Numbers>False</Store_Serial_Numbers>
<Sales_Order>0</Sales_Order>
<Purchase_Order>0</Purchase_Order>
<Transfer_Requests>0</Transfer_Requests>
<Transfer_Requests_Out>0</Transfer_Requests_Out>
<Vat_Rate>Normal Vat</Vat_Rate>
<Label_Quantity>0</Label_Quantity>
<Auto_Calculation_On>Never</Auto_Calculation_On>
<Markup_Cost_To_Use>Average Cost</Markup_Cost_To_Use>
<Reporting_Item/>
<Reporting_Factor>0</Reporting_Factor>
<Modules_Onhold/>
<Web_Item>False</Web_Item>
<Regular_Supplier/>
<MDR_Supplier_ID/>
<Suppliers/>
<Supplier_Item_Code/>
<Style/>
<Colour_Number>0</Colour_Number>
<Size_Number>0</Size_Number>
<New_Prices>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
  <New_Price>0</New_Price>
</New_Prices>
<Work_In_Progress_Quantity>0</Work_In_Progress_Quantity>
<Extended_Description/>
<Notes/>
<Picture/>
<Picture_Name/>
<Created>2018-10-20 08:19:01</Created>
<Modified>2018-10-20 09:28:02</Modified>
```

```
      <Cashier>1</Cashier>
      <Boxes>-1</Boxes>
      <Maximum_Discount>0</Maximum_Discount>
      <Is_Section7_Exempt>False</Is_Section7_Exempt>
      <Disallow_Decimals>True</Disallow_Decimals>
      <Unit_Of_Measure>Units</Unit_Of_Measure>
      <Line_Colour_Type>0</Line_Colour_Type>
      <Is_Scale_Item>False</Is_Scale_Item>
      <Status/>
      <Ordering_Method>Replenishment</Ordering_Method>
      <Order_Formula_Number>-1</Order_Formula_Number>
      <Override_GRV_Labels>False</Override_GRV_Labels>
      <Override_GRV_Label_Quantity>0</Override_GRV_Label_Quantity>
      <ABC_Classification/>
      <ABC_Classification_GP_Percentage>0</ABC_Classification_GP_Percentage>
      <Onhand>-1</Onhand>
      <Discount_In_Modules/>
      <Shelf_Life>0</Shelf_Life>
      <Last_Stock_Take>1899-12-30</Last_Stock_Take>
      <Document_Source>5</Document_Source>
      <Enable_Associated_Items>-1</Enable_Associated_Items>
      <Exclude_GRV_Extra>False</Exclude_GRV_Extra>
      <Exclude_Selling_Value>-1</Exclude_Selling_Value>
      <Is_Batch_Control>False</Is_Batch_Control>
      <Default_Line_Sales_Representative>0</Default_Line_Sales_Representative>
    </Stock>
    <Stock>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Stock":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
```

```
    "Company_Store_ID":"",
    "Company_Code":"001",
    "Company_Name":"test",
    "Company_Address1":"",
    "Company_Address2":"",
    "Company_Address3":"",
    "Company_Address4":"",
    "Company_Telephone1":"",
    "Company_Telephone2":"",
    "Company_Fax":"",
    "Company_Email":"",
    "Company_Tax":"",
    "Company_Registration_Number":"",
    "Company_Customs_Code":""
},
"Stock_Master":[
    {
        "Stock_Code":"123123",
        "Barcode":"123123",
        "Multiple_Barcodes":[

        ],
        "General_Code":"",
        "Description":"",
        "Alternative_Description":"",
        "Major_Department":"001",
        "Minor_Department":"0001",
        "Category":"",
        "Range":"",
        "Item_Category":"Stock Item",
        "Use_Fixed_Cost":false,
        "Cost_As_Percentage_Of_Sell_Price":0,
        "Pack_Size":0,
        "Pack_Description":"",
        "Bin_Location":"",
        "DATE_LastMoved":"2018-10-20",
        "DATE_LastPurchased":"1899-12-30",
        "DATE_LastSold":"2018-10-20",
        "Date_LastTransferred":"1899-12-30",
        "Previous_SellingPrice":0,
        "Previous_SellingPrice_Date":"2018-10-20",
        "Is_Sell_Price_Inclusive":true,
        "Sell_Prices":[
            {
                "Index":1,
                "Inclusive":0,
```

```
            "Exclusive":0
         },
         {
            "Index":2,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":3,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":4,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":5,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":6,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":7,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":8,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":9,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":10,
            "Inclusive":0,
            "Exclusive":0
         }
      ],
      "LayBye_Onhand":0,
```

```
"Average_Cost":0,
"Latest_Cost":0,
"Base_Cost":0,
"Previous_Cost":0,
"Highest_Cost":0,
"Minimum_Level":0,
"Maximum_Level":0,
"ReOrder_Level":0,
"ReOrder_Quantity":0,
"Store_Serial_Numbers":false,
"Sales_Order":0,
"Purchase_Order":0,
"Transfer_Requests":0,
"Transfer_Requests_Out":0,
"Vat_Rate":"Normal Vat",
"Label_Quantity":0,
"Auto_Calculation_On":"Never",
"Markup_Cost_To_Use":"Average Cost",
"Reporting_Item":"",
"Reporting_Factor":0,
"Modules_Onhold":[
   "Credit Notes",
   "Invoices and\/or Recurring Charges"
],
"Web_Item":false,
"Regular_Supplier":"",
"MDR_Supplier_ID":"",
"Suppliers":[
   {
      "Supplier_Account":"TEST",
      "Supplier_Code":"123",
      "Main_Supplier":false
   }
],
"Supplier_Item_Code":"",
"Style":"",
"Colour_Number":0,
"Size_Number":0,
"New_Prices":[
   0,
   0,
   0,
   0,
   0,
   0,
   0,
```

```
            0,
            0,
            0
        ],
        "Work_In_Progress_Quantity":0,
        "Extended_Description":"",
        "Notes":"",
        "Picture":"",
        "Picture_Name":"",
        "Created":"2018-10-20 08:19:01",
        "Modified":"2018-12-28 09:54:09",
        "Cashier":1,
        "Boxes":-1,
        "Maximum_Discount":0,
        "Is_Section7_Exempt":false,
        "Disallow_Decimals":true,
        "Unit_Of_Measure":"Units",
        "Line_Colour_Type":0,
        "Is_Scale_Item":false,
        "Status":"",
        "Ordering_Method":"Replenishment",
        "Order_Formula_Number":-1,
        "Override_GRV_Labels":false,
        "Override_GRV_Label_Quantity":0,
        "ABC_Classification":"",
        "ABC_Classification_GP_Percentage":0,
        "Onhand":-1,
        "Discount_In_Modules":[
            "Credit Notes",
            "Invoices and\/or Recurring Charges"
        ],
        "Shelf_Life":0,
        "Last_Stock_Take":"1899-12-30",
        "Document_Source":5,
        "Enable_Associated_Items":0,
        "Exclude_GRV_Extra":false,
        "Exclude_Selling_Value":0,
        "Is_Batch_Control":false,
        "Default_Line_Sales_Representative":0
    },
    {
        "Stock_Code":"2",
        "Barcode":"2",
        "Multiple_Barcodes":[

        ],
```

```
"General_Code":"",
"Description":"",
"Alternative_Description":"",
"Major_Department":"001",
"Minor_Department":"0001",
"Category":"",
"Range":"",
"Item_Category":"Stock Item",
"Use_Fixed_Cost":false,
"Cost_As_Percentage_Of_Sell_Price":0,
"Pack_Size":0,
"Pack_Description":"",
"Bin_Location":"",
"DATE_LastMoved":"1899-12-30",
"DATE_LastPurchased":"1899-12-30",
"DATE_LastSold":"1899-12-30",
"Date_LastTransferred":"1899-12-30",
"Previous_SellingPrice":0,
"Previous_SellingPrice_Date":"1899-12-30",
"Is_Sell_Price_Inclusive":true,
"Sell_Prices":[
   {
      "Index":1,
      "Inclusive":0,
      "Exclusive":0
   },
   {
      "Index":2,
      "Inclusive":0,
      "Exclusive":0
   },
   {
      "Index":3,
      "Inclusive":0,
      "Exclusive":0
   },
   {
      "Index":4,
      "Inclusive":0,
      "Exclusive":0
   },
   {
      "Index":5,
      "Inclusive":0,
      "Exclusive":0
   },
   {
```

```
            "Index":6,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":7,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":8,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":9,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":10,
            "Inclusive":0,
            "Exclusive":0
         }
      ],
      "LayBye_Onhand":0,
      "Average_Cost":0,
      "Latest_Cost":0,
      "Base_Cost":0,
      "Previous_Cost":0,
      "Highest_Cost":0,
      "Minimum_Level":0,
      "Maximum_Level":0,
      "ReOrder_Level":0,
      "ReOrder_Quantity":0,
      "Store_Serial_Numbers":false,
      "Sales_Order":0,
      "Purchase_Order":0,
      "Transfer_Requests":0,
      "Transfer_Requests_Out":0,
      "Vat_Rate":"Normal Vat",
      "Label_Quantity":0,
      "Auto_Calculation_On":"Never",
      "Markup_Cost_To_Use":"Average Cost",
      "Reporting_Item":"",
      "Reporting_Factor":0,
      "Modules_Onhold":[
```

```
],
"Web_Item":false,
"Regular_Supplier":"",
"MDR_Supplier_ID":"",
"Suppliers":[

],
"Supplier_Item_Code":"",
"Style":"",
"Colour_Number":0,
"Size_Number":0,
"New_Prices":[
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0
],
"Work_In_Progress_Quantity":0,
"Extended_Description":"",
"Notes":"",
"Picture":"",
"Picture_Name":"",
"Created":"2018-12-28 08:40:02",
"Modified":"2018-12-28 08:40:02",
"Cashier":1,
"Boxes":0,
"Maximum_Discount":0,
"Is_Section7_Exempt":false,
"Disallow_Decimals":true,
"Unit_Of_Measure":"Units",
"Line_Colour_Type":0,
"Is_Scale_Item":false,
"Status":"",
"Ordering_Method":"Replenishment",
"Order_Formula_Number":-1,
"Override_GRV_Labels":false,
"Override_GRV_Label_Quantity":0,
"ABC_Classification":"",
"ABC_Classification_GP_Percentage":0,
```

```
    "Onhand":0,
    "Discount_In_Modules":[

    ],
    "Shelf_Life":0,
    "Last_Stock_Take":"1899-12-30",
    "Document_Source":0,
    "Enable_Associated_Items":-1,
    "Exclude_GRV_Extra":false,
    "Exclude_Selling_Value":-1,
    "Is_Batch_Control":false,
    "Default_Line_Sales_Representative":0
},
{
    "Stock_Code":"3",
    "Barcode":"3",
    "Multiple_Barcodes":[

    ],
    "General_Code":"",
    "Description":"",
    "Alternative_Description":"",
    "Major_Department":"001",
    "Minor_Department":"0001",
    "Category":"",
    "Range":"",
    "Item_Category":"Stock Item",
    "Use_Fixed_Cost":false,
    "Cost_As_Percentage_Of_Sell_Price":0,
    "Pack_Size":0,
    "Pack_Description":"",
    "Bin_Location":"",
    "DATE_LastMoved":"1899-12-30",
    "DATE_LastPurchased":"1899-12-30",
    "DATE_LastSold":"1899-12-30",
    "Date_LastTransferred":"1899-12-30",
    "Previous_SellingPrice":0,
    "Previous_SellingPrice_Date":"1899-12-30",
    "Is_Sell_Price_Inclusive":true,
    "Sell_Prices":[
        {
            "Index":1,
            "Inclusive":0,
            "Exclusive":0
        },
        {
            "Index":2,
```

```
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":3,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":4,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":5,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":6,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":7,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":8,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":9,
            "Inclusive":0,
            "Exclusive":0
         },
         {
            "Index":10,
            "Inclusive":0,
            "Exclusive":0
         }
      ],
      "LayBye_Onhand":0,
      "Average_Cost":0,
      "Latest_Cost":0,
      "Base_Cost":0,
      "Previous_Cost":0,
```

```
"Highest_Cost":0,
"Minimum_Level":0,
"Maximum_Level":0,
"ReOrder_Level":0,
"ReOrder_Quantity":0,
"Store_Serial_Numbers":false,
"Sales_Order":0,
"Purchase_Order":0,
"Transfer_Requests":0,
"Transfer_Requests_Out":0,
"Vat_Rate":"Normal Vat",
"Label_Quantity":0,
"Auto_Calculation_On":"Never",
"Markup_Cost_To_Use":"Average Cost",
"Reporting_Item":"",
"Reporting_Factor":0,
"Modules_Onhold":[

],
"Web_Item":false,
"Regular_Supplier":"",
"MDR_Supplier_ID":"",
"Suppliers":[

],
"Supplier_Item_Code":"",
"Style":"",
"Colour_Number":0,
"Size_Number":0,
"New_Prices":[
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0,
   0
],
"Work_In_Progress_Quantity":0,
"Extended_Description":"",
"Notes":"",
"Picture":"",
"Picture_Name":"",
```

```
            "Created":"2018-12-28 08:40:07",
            "Modified":"2018-12-28 08:40:07",
            "Cashier":1,
            "Boxes":0,
            "Maximum_Discount":0,
            "Is_Section7_Exempt":false,
            "Disallow_Decimals":true,
            "Unit_Of_Measure":"Units",
            "Line_Colour_Type":0,
            "Is_Scale_Item":false,
            "Status":"",
            "Ordering_Method":"Replenishment",
            "Order_Formula_Number":-1,
            "Override_GRV_Labels":false,
            "Override_GRV_Label_Quantity":0,
            "ABC_Classification":"",
            "ABC_Classification_GP_Percentage":0,
            "Onhand":0,
            "Discount_In_Modules":[

            ],
            "Shelf_Life":0,
            "Last_Stock_Take":"1899-12-30",
            "Document_Source":0,
            "Enable_Associated_Items":-1,
            "Exclude_GRV_Extra":false,
            "Exclude_Selling_Value":-1,
            "Is_Batch_Control":false,
            "Default_Line_Sales_Representative":0
         }
      ]
   }
}
```

Import Export Object: Stock Active Selling Price

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Stock_ActiveSellingPrice>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Stock_ActiveSellingPrice>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <StockDebtorActivePricing>
    <Stock_Code>123123</Stock_Code>
    <Price_Inclusive>115</Price_Inclusive>
    <Price_Exclusive>100</Price_Exclusive>
    <Line_Discount_Percentage>10</Line_Discount_Percentage>
    <Price_Type>1</Price_Type>
   </StockDebtorActivePricing>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Stock_ActiveSellingPrice":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
```

```
    }
}
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Stock_Active_Selling_Price":[
            {
               "Stock_Code":"123123",
               "Price_Inclusive":115,
               "Price_Exclusive":100,
               "Line_Discount_Percentage":10,
               "Price_Type":1
            }
         ]
      }
   }
}
```

Import Export Object: Active Till Shift Number

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_TillShiftNo>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_TillShiftNo>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <TillShiftNo>
    <TillNo>1</TillNo>
    <ShiftNo>1</ShiftNo>
    <TradingDate>2018-12-29</TradingDate>
   </TillShiftNo>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_TillShiftNo":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
```

```
    }
}
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Till_ShiftNo":[
            {
               "TillNo":1,
               "ShiftNo":1,
               "TradingDate":"2018-12-29"
            }
         ]
      }
   }
}
```

Import Export Object: Processing Invoice

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
  <IQ_API_Request_Document_Invoice>
    <IQ_Company_Number>001</IQ_Company_Number>
    <IQ_Terminal_Number>1</IQ_Terminal_Number>
    <IQ_User_Number>99</IQ_User_Number>
    <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
    <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
  </IQ_API_Request_Document_Invoice>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
    <IQ_API_Error>
        <IQ_Error_Code>0</IQ_Error_Code>
    </IQ_API_Error>
    <IQ_API_Result_Data>
        <IQ_Root_XML>
            <IQ_Identification_Info>
                <Company_Store_ID/>
                <Company_Code>001</Company_Code>
                <Company_Name>test</Company_Name>
                <Company_Address1/>
                <Company_Address2/>
                <Company_Address3/>
                <Company_Address4/>
                <Company_Telephone1/>
                <Company_Telephone2/>
                <Company_Fax/>
                <Company_Email/>
                <Company_Tax/>
                <Company_Registration_Number/>
                <Company_Customs_Code/>
            </IQ_Identification_Info>
            <Invoice>
                <Document>
                    <Document_Number>INV0</Document_Number>
                    <Processed_Document_Number/>
                    <Delivery_Address_Information>
                        <Delivery_Address_Detail/>
                        <Delivery_Address_Detail/>
                       <Delivery_Address_Detail/>
                        <Delivery_Address_Detail/>
                    </Delivery_Address_Information>
                    <Email_Address>danie@simplydot.co.za</Email_Address>
                    <Order_Number/>
                    <Delivery_Method/>
                    <Delivery_Note_Number/>
                    <Total_Vat>2.87</Total_Vat>
                    <Discount_Percentage>0</Discount_Percentage>
                    <Discount_Type>Percentage</Discount_Type>
                    <Discount_Amount>0</Discount_Amount>
                    <Long_Description/>
                    <Document_Total>22</Document_Total>
                    <Total_Number_Of_Items>1</Total_Number_Of_Items>
                    <Document_Description/>
```

```
        <Print_Layout>1</Print_Layout>
        <Warehouse/>
        <Cashier_Number>1</Cashier_Number>
        <Till_Number>1</Till_Number>
        <Document_Includes_VAT>True</Document_Includes_VAT>
        <Currency>ZAR</Currency>
        <Currency_Rate>1</Currency_Rate>
        <Internal_Order_Number/>
        <Store_Department/>
        <Document_Terms>Not Applicable</Document_Terms>
        <Telephone_Number/>
        <Extra_Charges_Information>
            <Extra_Charge_Information_1>
                <Extra_Charge_Description/>
                <Extra_Charge_Amount>0</Extra_Charge_Amount>
            </Extra_Charge_Information_1>
            <Extra_Charge_Information_2>
                <Extra_Charge_Description/>
                <Extra_Charge_Amount>0</Extra_Charge_Amount>
            </Extra_Charge_Information_2>
            <Extra_Charge_Information_3>
                <Extra_Charge_Description/>
                <Extra_Charge_Amount>0</Extra_Charge_Amount>
            </Extra_Charge_Information_3>
            <Extra_Charge_Information_4>
                <Extra_Charge_Description/>
                <Extra_Charge_Amount>0</Extra_Charge_Amount>
            </Extra_Charge_Information_4>
        </Extra_Charges_Information>
        <Loyalty_Account/>
        <Debtor_Account>ASDASD</Debtor_Account>
        <Sales_Representative_Number>1</Sales_Representative_Number>
        <Invoice_Date>2018-10-20</Invoice_Date>
        <Picker_Number>0</Picker_Number>
        <Date_Picked_On>2018-10-20</Date_Picked_On>
    </Document>
    <Items>
        <Item>
            <Stock_Code>123123</Stock_Code>
            <Comment/>
            <Quantity>1</Quantity>
            <Volumetrics>
                <Units>0</Units>
                <Volume_Length>0</Volume_Length>
                <Volume_Width>0</Volume_Width>
                <Volume_Height>0</Volume_Height>
                <Volume_Quantity>1</Volume_Quantity>
                <Volume_Value>0</Volume_Value>
                <Volume_Rounding>0</Volume_Rounding>
            </Volumetrics>
            <Item_Price_Inclusive>22</Item_Price_Inclusive>
            <Item_Price_Exclusive>19.1304347826087</Item_Price_Exclusive>
            <Discount_Percentage>0</Discount_Percentage>
            <Line_Total_Inclusive>22</Line_Total_Inclusive>
            <Line_Total_Exclusive>19.1304347826087</Line_Total_Exclusive>
            <Custom_Cost>0</Custom_Cost>
            <List_Price>0</List_Price>
            <Serials/>
        </Item>
    </Items>
```

```
            </Invoice>
         </IQ_Root_XML>
      </IQ_API_Result_Data>
</IQ_API_Result>
```


```
{
   "IQ_API":{
      "IQ_API_Request_Document_Invoice":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
            "Company_Address1":"",
            "Company_Address2":"",
            "Company_Address3":"",
            "Company_Address4":"",
            "Company_Telephone1":"",
            "Company_Telephone2":"",
            "Company_Fax":"",
            "Company_Email":"",
            "Company_Tax":"",
            "Company_Registration_Number":"",
            "Company_Customs_Code":""
         },
         "Processing_Documents_Invoice":[
            {
               "Document":{
                  "Document_Number":"INV0",
                  "Processed_Document_Number":"",
                  "Delivery_Address_Information":[
                     "",
                     "",
                     "",
                     ""
                  ],
```

```
"Email_Address":"danie@simplydot.co.za",
"Order_Number":"",
"Delivery_Method":"",
"Delivery_Note_Number":"",
"Total_Vat":2.87,
"Discount_Percentage":0,
"Discount_Type":"Percentage",
"Discount_Amount":0,
"Long_Description":"",
"Document_Total":22,
"Total_Number_Of_Items":1,
"Document_Description":"",
"Print_Layout":1,
"Warehouse":"",
"Cashier_Number":1,
"Till_Number":1,
"Document_Includes_VAT":true,
"Currency":"ZAR",
"Currency_Rate":1,
"Internal_Order_Number":"",
"Store_Department":"",
"Document_Terms":"Not Applicable",
"Telephone_Number":"",
"Extra_Charges_Information":[
   {
      "Extra_Charge_Description":"",
      "Extra_Charge_Amount":0
   },
   {
      "Extra_Charge_Description":"",
      "Extra_Charge_Amount":0
   },
   {
      "Extra_Charge_Description":"",
      "Extra_Charge_Amount":0
   },
   {
      "Extra_Charge_Description":"",
      "Extra_Charge_Amount":0
   }
],
"Loyalty_Account":"",
"Debtor_Account":"ASDASD",
"Sales_Representative_Number":1,
"Invoice_Date":"2018-10-20",
"Picker_Number":0,
"Date_Picked_On":"2018-10-20"
```

```
            },
            "Items":[
               {
                  "Stock_Code":"123123",
                  "Comment":"",
                  "Quantity":1,
                  "Volumetrics":{
                     "Units":0,
                     "Volume_Length":0,
                     "Volume_Width":0,
                     "Volume_Height":0,
                     "Volume_Quantity":1,
                     "Volume_Value":0,
                     "Volume_Rounding":0
                  },
                  "Item_Price_Inclusive":22,
                  "Item_Price_Exclusive":19.1304347826087,
                  "Discount_Percentage":0,
                  "Line_Total_Inclusive":22,
                  "Line_Total_Exclusive":19.1304347826087,
                  "Custom_Cost":0,
                  "List_Price":0,
                  "Serials":[

                  ]
               }
            ]
         }
      ]
   }
}
```

Import Export Object: Processing Purchase Order

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Document_Purchase_Order>
  <IQ_Company_Number>001</IQ_Company_Number>
|  <IQ_Terminal_Number>1</IQ_Terminal_Number>
|  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Document_Purchase_Order>
</IQ_API>


<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Purchase_Order>
    <Document>
     <Document_Number>PUR0</Document_Number>
     <Processed_Document_Number/>
     <Delivery_Address_Information>
      <Delivery_Address_Detail>123</Delivery_Address_Detail>
      <Delivery_Address_Detail>123</Delivery_Address_Detail>
      <Delivery_Address_Detail>123</Delivery_Address_Detail>
      <Delivery_Address_Detail>123</Delivery_Address_Detail>
     </Delivery_Address_Information>
     <Email_Address>444</Email_Address>
     <Order_Number/>
     <Delivery_Method/>
     <Delivery_Note_Number/>
     <Total_Vat>1.7</Total_Vat>
     <Discount_Percentage>0</Discount_Percentage>
     <Discount_Type>Percentage</Discount_Type>
     <Discount_Amount>0</Discount_Amount>
     <Long_Description/>
     <Document_Total>13</Document_Total>
```

```
<Total_Number_Of_Items>5</Total_Number_Of_Items>
<Document_Description/>
<Print_Layout>1</Print_Layout>
<Warehouse/>
<Cashier_Number>1</Cashier_Number>
<Till_Number>1</Till_Number>
<Document_Includes_VAT>True</Document_Includes_VAT>
<Currency>ZAR</Currency>
<Currency_Rate>1</Currency_Rate>
<Internal_Order_Number/>
<Store_Department/>
<Document_Terms>Not Applicable</Document_Terms>
<Telephone_Number>444</Telephone_Number>
<Extra_Charges_Information>
 <Extra_Charge_Information_1>
  <Extra_Charge_Description/>
  <Extra_Charge_Amount>0</Extra_Charge_Amount>
 </Extra_Charge_Information_1>
 <Extra_Charge_Information_2>
  <Extra_Charge_Description/>
  <Extra_Charge_Amount>0</Extra_Charge_Amount>
 </Extra_Charge_Information_2>
 <Extra_Charge_Information_3>
  <Extra_Charge_Description/>
  <Extra_Charge_Amount>0</Extra_Charge_Amount>
 </Extra_Charge_Information_3>
 <Extra_Charge_Information_4>
  <Extra_Charge_Description/>
  <Extra_Charge_Amount>0</Extra_Charge_Amount>
 </Extra_Charge_Information_4>
</Extra_Charges_Information>
<Creditor_Account>TEST</Creditor_Account>
<Order_Information>
 <Order_Date>2018-12-28</Order_Date>
 <Expected_Date>2018-12-28</Expected_Date>
 <Credit_Approver_Number>0</Credit_Approver_Number>
</Order_Information>
</Document>
<Items>
 <Item>
  <Stock_Code>123123</Stock_Code>
  <Comment/>
  <Quantity>2</Quantity>
  <Volumetrics>
   <Units>0</Units>
   <Volume_Length>0</Volume_Length>
   <Volume_Width>0</Volume_Width>
   <Volume_Height>0</Volume_Height>
   <Volume_Quantity>1</Volume_Quantity>
   <Volume_Value>0</Volume_Value>
   <Volume_Rounding>0</Volume_Rounding>
  </Volumetrics>
  <Item_Price_Inclusive>2</Item_Price_Inclusive>
  <Item_Price_Exclusive>1.73913</Item_Price_Exclusive>
  <Discount_Percentage>0</Discount_Percentage>
  <Line_Total_Inclusive>4</Line_Total_Inclusive>
```

```
      <Line_Total_Exclusive>3.478261</Line_Total_Exclusive>
      <Custom_Cost>0</Custom_Cost>
      <List_Price>0</List_Price>
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
     <Item>
      <Stock_Code>3</Stock_Code>
      <Comment/>
      <Quantity>3</Quantity>
      <Volumetrics>
       <Units>0</Units>
       <Volume_Length>0</Volume_Length>
       <Volume_Width>0</Volume_Width>
       <Volume_Height>0</Volume_Height>
       <Volume_Quantity>1</Volume_Quantity>
       <Volume_Value>0</Volume_Value>
       <Volume_Rounding>0</Volume_Rounding>
      </Volumetrics>
      <Item_Price_Inclusive>3</Item_Price_Inclusive>
      <Item_Price_Exclusive>2.608696</Item_Price_Exclusive>
      <Discount_Percentage>0</Discount_Percentage>
      <Line_Total_Inclusive>9</Line_Total_Inclusive>
      <Line_Total_Exclusive>7.826087</Line_Total_Exclusive>
      <Custom_Cost>0</Custom_Cost>
      <List_Price>0</List_Price>
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
    </Items>
   </Purchase_Order>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Document_Purchase_Order":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
```

```
    "Company_Name":"test",
    "Company_Address1":"",
    "Company_Address2":"",
    "Company_Address3":"",
    "Company_Address4":"",
    "Company_Telephone1":"",
    "Company_Telephone2":"",
    "Company_Fax":"",
    "Company_Email":"",
    "Company_Tax":"",
    "Company_Registration_Number":"",
    "Company_Customs_Code":""
},
"Processing_Documents_Purchase_Order":[
    {
        "Document":{
            "Document_Number":"PUR0",
            "Processed_Document_Number":"",
            "Delivery_Address_Information":[
                "123",
                "123",
                "123",
                "123"
            ],
            "Email_Address":"444",
            "Order_Number":"",
            "Delivery_Method":"",
            "Delivery_Note_Number":"",
            "Total_Vat":1.7,
            "Discount_Percentage":0,
            "Discount_Type":"Percentage",
            "Discount_Amount":0,
            "Long_Description":"",
            "Document_Total":13,
            "Total_Number_Of_Items":5,
            "Document_Description":"",
            "Print_Layout":1,
            "Warehouse":"",
            "Cashier_Number":1,
            "Till_Number":1,
            "Document_Includes_VAT":true,
            "Currency":"ZAR",
            "Currency_Rate":1,
            "Internal_Order_Number":"",
            "Store_Department":"",
            "Document_Terms":"Not Applicable",
```

```
    "Telephone_Number":"444",
    "Extra_Charges_Information":[
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       }
    ],
    "Creditor_Account":"TEST",
    "Order_Information":{
       "Order_Date":"2018-12-28",
       "Expected_Date":"2018-12-28",
       "Credit_Approver_Number":0
    }
 },
 "Items":[
    {
       "Stock_Code":"123123",
       "Comment":"",
       "Quantity":2,
       "Volumetrics":{
          "Units":0,
          "Volume_Length":0,
          "Volume_Width":0,
          "Volume_Height":0,
          "Volume_Quantity":1,
          "Volume_Value":0,
          "Volume_Rounding":0
       },
       "Item_Price_Inclusive":2,
       "Item_Price_Exclusive":1.73913,
       "Discount_Percentage":0,
       "Line_Total_Inclusive":4,
       "Line_Total_Exclusive":3.478261,
       "Custom_Cost":0,
       "List_Price":0,
       "Invoiced_Quantity":0
    },
```

```
                {
                   "Stock_Code":"3",
                   "Comment":"",
                   "Quantity":3,
                   "Volumetrics":{
                      "Units":0,
                      "Volume_Length":0,
                      "Volume_Width":0,
                      "Volume_Height":0,
                      "Volume_Quantity":1,
                      "Volume_Value":0,
                      "Volume_Rounding":0
                   },
                   "Item_Price_Inclusive":3,
                   "Item_Price_Exclusive":2.608696,
                   "Discount_Percentage":0,
                   "Line_Total_Inclusive":9,
                   "Line_Total_Exclusive":7.826087,
                   "Custom_Cost":0,
                   "List_Price":0,
                   "Invoiced_Quantity":0
                }
             ]
          }
       ]
    }
 }
}
```

Import Export Object: Processing Quote

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Document_Quote>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Document_Quote>
</IQ_API>

<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>

   </IQ_Identification_Info>
   <Quote>
    <Document>
     <Document_Number>QTE0</Document_Number>
     <Processed_Document_Number/>
     <Delivery_Address_Information>
      <Delivery_Address_Detail>del1</Delivery_Address_Detail>
      <Delivery_Address_Detail>2</Delivery_Address_Detail>
      <Delivery_Address_Detail>3</Delivery_Address_Detail>
      <Delivery_Address_Detail>4</Delivery_Address_Detail>
     </Delivery_Address_Information>
     <Email_Address>danie@simplydot.co.za</Email_Address>
     <Order_Number/>
     <Delivery_Method/>
     <Delivery_Note_Number/>
     <Total_Vat>251.22</Total_Vat>
     <Discount_Percentage>10</Discount_Percentage>
     <Discount_Type>Percentage</Discount_Type>
     <Discount_Amount>186.087</Discount_Amount>
     <Long_Description/>
     <Document_Total>1926</Document_Total>
```

```
 <Total_Number_Of_Items>46</Total_Number_Of_Items>
 <Document_Description/>
 <Print_Layout>1</Print_Layout>
 <Warehouse/>
 <Cashier_Number>1</Cashier_Number>
 <Till_Number>1</Till_Number>
 <Document_Includes_VAT>True</Document_Includes_VAT>
 <Currency>ZAR</Currency>
 <Currency_Rate>1</Currency_Rate>
 <Internal_Order_Number/>
 <Store_Department>DR_DEPT1</Store_Department>
 <Document_Terms>Not Applicable</Document_Terms>
 <Telephone_Number>021</Telephone_Number>
 <Extra_Charges_Information>
  <Extra_Charge_Information_1>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_1>
  <Extra_Charge_Information_2>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_2>
  <Extra_Charge_Information_3>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_3>
  <Extra_Charge_Information_4>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_4>
 </Extra_Charges_Information>
 <Debtor_Account>ASDASD</Debtor_Account>
 <Sales_Representative_Number>1</Sales_Representative_Number>
 <Order_Date>2018-12-28</Order_Date>
</Document>
<Items>
 <Item>
  <Stock_Code>123123</Stock_Code>
  <Comment/>
  <Quantity>1</Quantity>
  <Volumetrics>
   <Units>0</Units>
   <Volume_Length>0</Volume_Length>
   <Volume_Width>0</Volume_Width>
   <Volume_Height>0</Volume_Height>
   <Volume_Quantity>1</Volume_Quantity>
   <Volume_Value>0</Volume_Value>
   <Volume_Rounding>0</Volume_Rounding>
  </Volumetrics>
  <Item_Price_Inclusive>115</Item_Price_Inclusive>
  <Item_Price_Exclusive>100</Item_Price_Exclusive>
  <Discount_Percentage>0</Discount_Percentage>
  <Line_Total_Inclusive>115</Line_Total_Inclusive>
  <Line_Total_Exclusive>100</Line_Total_Exclusive>
  <Custom_Cost>0</Custom_Cost>
  <List_Price>115</List_Price>
```

```
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
     <Item>
      <Stock_Code>SERIAL</Stock_Code>
      <Comment/>
      <Quantity>45</Quantity>
      <Volumetrics>
       <Units>0</Units>
       <Volume_Length>0</Volume_Length>
       <Volume_Width>0</Volume_Width>
       <Volume_Height>0</Volume_Height>
       <Volume_Quantity>1</Volume_Quantity>
       <Volume_Value>0</Volume_Value>
       <Volume_Rounding>0</Volume_Rounding>
      </Volumetrics>
      <Item_Price_Inclusive>45</Item_Price_Inclusive>
      <Item_Price_Exclusive>39.1304347826087</Item_Price_Exclusive>
      <Discount_Percentage>0</Discount_Percentage>
      <Line_Total_Inclusive>2025</Line_Total_Inclusive>
      <Line_Total_Exclusive>1760.86956521739</Line_Total_Exclusive>
      <Custom_Cost>0</Custom_Cost>
      <List_Price>0</List_Price>
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
    </Items>
   </Quote>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Document_Quote":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
      "IQ_Root_JSON":{
         "IQ_Identification_Info":{
            "Company_Store_ID":"",
            "Company_Code":"001",
            "Company_Name":"test",
```

```
   "Company_Address1":"",
   "Company_Address2":"",
   "Company_Address3":"",
   "Company_Address4":"",
   "Company_Telephone1":"",
   "Company_Telephone2":"",
   "Company_Fax":"",
   "Company_Email":"",
   "Company_Tax":"",
   "Company_Registration_Number":"",
   "Company_Customs_Code":""
},
"Processing_Documents_Quote":[
   {
      "Document":{
         "Document_Number":"QTE0",
         "Processed_Document_Number":"",
         "Delivery_Address_Information":[
            "del1",
            "2",
            "3",
            "4"
         ],
         "Email_Address":"danie@simplydot.co.za",
         "Order_Number":"",
         "Delivery_Method":"",
         "Delivery_Note_Number":"",
         "Total_Vat":251.22,
         "Discount_Percentage":10,
         "Discount_Type":"Percentage",
         "Discount_Amount":186.087,
         "Long_Description":"",
         "Document_Total":1926,
         "Total_Number_Of_Items":46,
         "Document_Description":"",
         "Print_Layout":1,
         "Warehouse":"",
         "Cashier_Number":1,
         "Till_Number":1,
         "Document_Includes_VAT":true,
         "Currency":"ZAR",
         "Currency_Rate":1,
         "Internal_Order_Number":"",
         "Store_Department":"DR_DEPT1",
         "Document_Terms":"Not Applicable",
         "Telephone_Number":"021",
```

```
    "Extra_Charges_Information":[
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       }
    ],
    "Debtor_Account":"ASDASD",
    "Sales_Representative_Number":1,
    "Order_Date":"2018-12-28"
 },
 "Items":[
    {
       "Stock_Code":"123123",
       "Comment":"",
       "Quantity":1,
       "Volumetrics":{
          "Units":0,
          "Volume_Length":0,
          "Volume_Width":0,
          "Volume_Height":0,
          "Volume_Quantity":1,
          "Volume_Value":0,
          "Volume_Rounding":0
       },
       "Item_Price_Inclusive":115,
       "Item_Price_Exclusive":100,
       "Discount_Percentage":0,
       "Line_Total_Inclusive":115,
       "Line_Total_Exclusive":100,
       "Custom_Cost":0,
       "List_Price":115,
       "Invoiced_Quantity":0
    },
    {
       "Stock_Code":"SERIAL",
       "Comment":"",
```

```
                    "Quantity":45,
                    "Volumetrics":{
                       "Units":0,
                       "Volume_Length":0,
                       "Volume_Width":0,
                       "Volume_Height":0,
                       "Volume_Quantity":1,
                       "Volume_Value":0,
                       "Volume_Rounding":0
                    },
                    "Item_Price_Inclusive":45,
                    "Item_Price_Exclusive":39.1304347826087,
                    "Discount_Percentage":0,
                    "Line_Total_Inclusive":2025,
                    "Line_Total_Exclusive":1760.86956521739,
                    "Custom_Cost":0,
                    "List_Price":0,
                    "Invoiced_Quantity":0
                 }
              ]
           }
        ]
     }
  }
}
```

Import Export Object: Processing Sales Order

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API>
 <IQ_API_Request_Document_Sales_Order>
  <IQ_Company_Number>001</IQ_Company_Number>
  <IQ_Terminal_Number>1</IQ_Terminal_Number>
  <IQ_User_Number>99</IQ_User_Number>
  <IQ_User_Password>C89ECE949D7A657B4508788FD2496088EABFD870</IQ_User_Password>
  <IQ_Partner_Passphrase>743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA</IQ_Partner_Passphrase>
 </IQ_API_Request_Document_Sales_Order>
</IQ_API>
```

```
<?xml version="1.0" encoding="windows-1252"?>
<IQ_API_Result>
 <IQ_API_Error>
  <IQ_Error_Code>0</IQ_Error_Code>
 </IQ_API_Error>
 <IQ_API_Result_Data>
  <IQ_Root_XML>
   <IQ_Identification_Info>
    <Company_Store_ID/>
    <Company_Code>001</Company_Code>
    <Company_Name>test</Company_Name>
    <Company_Address1/>
    <Company_Address2/>
    <Company_Address3/>
    <Company_Address4/>
    <Company_Telephone1/>
    <Company_Telephone2/>
    <Company_Fax/>
    <Company_Email/>
    <Company_Tax/>
    <Company_Registration_Number/>
    <Company_Customs_Code/>
   </IQ_Identification_Info>
   <Sales_Order>
    <Document>
     <Document_Number>SAL0</Document_Number>
     <Processed_Document_Number/>
     <Delivery_Address_Information>
      <Delivery_Address_Detail>del1</Delivery_Address_Detail>
      <Delivery_Address_Detail>2</Delivery_Address_Detail>
      <Delivery_Address_Detail>3</Delivery_Address_Detail>
      <Delivery_Address_Detail>4</Delivery_Address_Detail>
     </Delivery_Address_Information>
     <Email_Address>danie@simplydot.co.za</Email_Address>
     <Order_Number/>
     <Delivery_Method/>
     <Delivery_Note_Number/>
     <Total_Vat>39.91</Total_Vat>
     <Discount_Percentage>10</Discount_Percentage>
     <Discount_Type>Percentage</Discount_Type>
     <Discount_Amount>29.5652</Discount_Amount>
     <Long_Description/>
```

```
 <Document_Total>306</Document_Total>
 <Total_Number_Of_Items>6</Total_Number_Of_Items>
 <Document_Description/>
 <Print_Layout>1</Print_Layout>
 <Warehouse/>
 <Cashier_Number>1</Cashier_Number>
 <Till_Number>1</Till_Number>
 <Document_Includes_VAT>True</Document_Includes_VAT>
 <Currency>ZAR</Currency>
 <Currency_Rate>1</Currency_Rate>
 <Internal_Order_Number/>
 <Store_Department/>
 <Document_Terms>Not Applicable</Document_Terms>
 <Telephone_Number>021</Telephone_Number>
 <Extra_Charges_Information>
  <Extra_Charge_Information_1>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_1>
  <Extra_Charge_Information_2>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_2>
  <Extra_Charge_Information_3>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_3>
  <Extra_Charge_Information_4>
   <Extra_Charge_Description/>
   <Extra_Charge_Amount>0</Extra_Charge_Amount>
  </Extra_Charge_Information_4>
 </Extra_Charges_Information>
 <Debtor_Account>ASDASD</Debtor_Account>
 <Sales_Representative_Number>1</Sales_Representative_Number>
 <Order_Information>
  <Order_Date>2018-12-28</Order_Date>
  <Expected_Date>2018-12-28</Expected_Date>
  <Credit_Approver_Number>0</Credit_Approver_Number>
 </Order_Information>
</Document>
<Items>
 <Item>
  <Stock_Code>123123</Stock_Code>
  <Comment/>
  <Quantity>1</Quantity>
  <Volumetrics>
   <Units>0</Units>
   <Volume_Length>0</Volume_Length>
   <Volume_Width>0</Volume_Width>
   <Volume_Height>0</Volume_Height>
   <Volume_Quantity>1</Volume_Quantity>
   <Volume_Value>0</Volume_Value>
   <Volume_Rounding>0</Volume_Rounding>
  </Volumetrics>
  <Item_Price_Inclusive>115</Item_Price_Inclusive>
  <Item_Price_Exclusive>100</Item_Price_Exclusive>
```

```
      <Discount_Percentage>0</Discount_Percentage>
      <Line_Total_Inclusive>115</Line_Total_Inclusive>
      <Line_Total_Exclusive>100</Line_Total_Exclusive>
      <Custom_Cost>0</Custom_Cost>
      <List_Price>115</List_Price>
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
     <Item>
      <Stock_Code>SERIAL</Stock_Code>
      <Comment/>
      <Quantity>5</Quantity>
      <Volumetrics>
       <Units>0</Units>
       <Volume_Length>0</Volume_Length>
       <Volume_Width>0</Volume_Width>
       <Volume_Height>0</Volume_Height>
       <Volume_Quantity>1</Volume_Quantity>
       <Volume_Value>0</Volume_Value>
       <Volume_Rounding>0</Volume_Rounding>
      </Volumetrics>
      <Item_Price_Inclusive>45</Item_Price_Inclusive>
      <Item_Price_Exclusive>39.130435</Item_Price_Exclusive>
      <Discount_Percentage>0</Discount_Percentage>
      <Line_Total_Inclusive>225</Line_Total_Inclusive>
      <Line_Total_Exclusive>195.652174</Line_Total_Exclusive>
      <Custom_Cost>0</Custom_Cost>
      <List_Price>0</List_Price>
      <Invoiced_Quantity>0</Invoiced_Quantity>
     </Item>
    </Items>
   </Sales_Order>
  </IQ_Root_XML>
 </IQ_API_Result_Data>
</IQ_API_Result>
```

```
{
   "IQ_API":{
      "IQ_API_Request_Document_Sales_Order":{
         "IQ_Company_Number":"001",
         "IQ_Terminal_Number":1,
         "IQ_User_Number":99,
         "IQ_User_Password":"C89ECE949D7A657B4508788FD2496088EABFD870",
         "IQ_Partner_Passphrase":"743B25C6C57BA9A4D02EBBAD9D11B9ADC47A1BCA"
      }
   }
}
```

```
{
   "IQ_API_Error":[
      {
         "IQ_Error_Code":0
      }
   ],
   "IQ_API_Result_Data":{
```

```
"IQ_Root_JSON":{
   "IQ_Identification_Info":{
      "Company_Store_ID":"",
      "Company_Code":"001",
      "Company_Name":"test",
      "Company_Address1":"",
      "Company_Address2":"",
      "Company_Address3":"",
      "Company_Address4":"",
      "Company_Telephone1":"",
      "Company_Telephone2":"",
      "Company_Fax":"",
      "Company_Email":"",
      "Company_Tax":"",
      "Company_Registration_Number":"",
      "Company_Customs_Code":""
   },
   "Processing_Documents_Sales_Order":[
      {
         "Document":{
            "Document_Number":"SAL0",
            "Processed_Document_Number":"",
            "Delivery_Address_Information":[
               "del1",
               "2",
               "3",
               "4"
            ],
            "Email_Address":"danie@simplydot.co.za",
            "Order_Number":"",
            "Delivery_Method":"",
            "Delivery_Note_Number":"",
            "Total_Vat":39.91,
            "Discount_Percentage":10,
            "Discount_Type":"Percentage",
            "Discount_Amount":29.5652,
            "Long_Description":"",
            "Document_Total":306,
            "Total_Number_Of_Items":6,
            "Document_Description":"",
            "Print_Layout":1,
            "Warehouse":"",
            "Cashier_Number":1,
            "Till_Number":1,
            "Document_Includes_VAT":true,
            "Currency":"ZAR",
```

```
    "Currency_Rate":1,
    "Internal_Order_Number":"",
    "Store_Department":"",
    "Document_Terms":"Not Applicable",
    "Telephone_Number":"021",
    "Extra_Charges_Information":[
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       },
       {
          "Extra_Charge_Description":"",
          "Extra_Charge_Amount":0
       }
    ],
    "Debtor_Account":"ASDASD",
    "Sales_Representative_Number":1,
    "Order_Information":{
       "Order_Date":"2018-12-28",
       "Expected_Date":"2018-12-28",
       "Credit_Approver_Number":0
    }
 },
 "Items":[
    {
       "Stock_Code":"123123",
       "Comment":"",
       "Quantity":1,
       "Volumetrics":{
          "Units":0,
          "Volume_Length":0,
          "Volume_Width":0,
          "Volume_Height":0,
          "Volume_Quantity":1,
          "Volume_Value":0,
          "Volume_Rounding":0
       },
       "Item_Price_Inclusive":115,
       "Item_Price_Exclusive":100,
       "Discount_Percentage":0,
```

```
                "Line_Total_Inclusive":115,
                "Line_Total_Exclusive":100,
                "Custom_Cost":0,
                "List_Price":115,
                "Invoiced_Quantity":0
            },
            {
                "Stock_Code":"SERIAL",
                "Comment":"",
                "Quantity":5,
                "Volumetrics":{
                    "Units":0,
                    "Volume_Length":0,
                    "Volume_Width":0,
                    "Volume_Height":0,
                    "Volume_Quantity":1,
                    "Volume_Value":0,
                    "Volume_Rounding":0
                },
                "Item_Price_Inclusive":45,
                "Item_Price_Exclusive":39.130435,
                "Discount_Percentage":0,
                "Line_Total_Inclusive":225,
                "Line_Total_Exclusive":195.652174,
                "Custom_Cost":0,
                "List_Price":0,
                "Invoiced_Quantity":0
            }
        ]
    }
]
}
}
}
```