TDOX WEB API

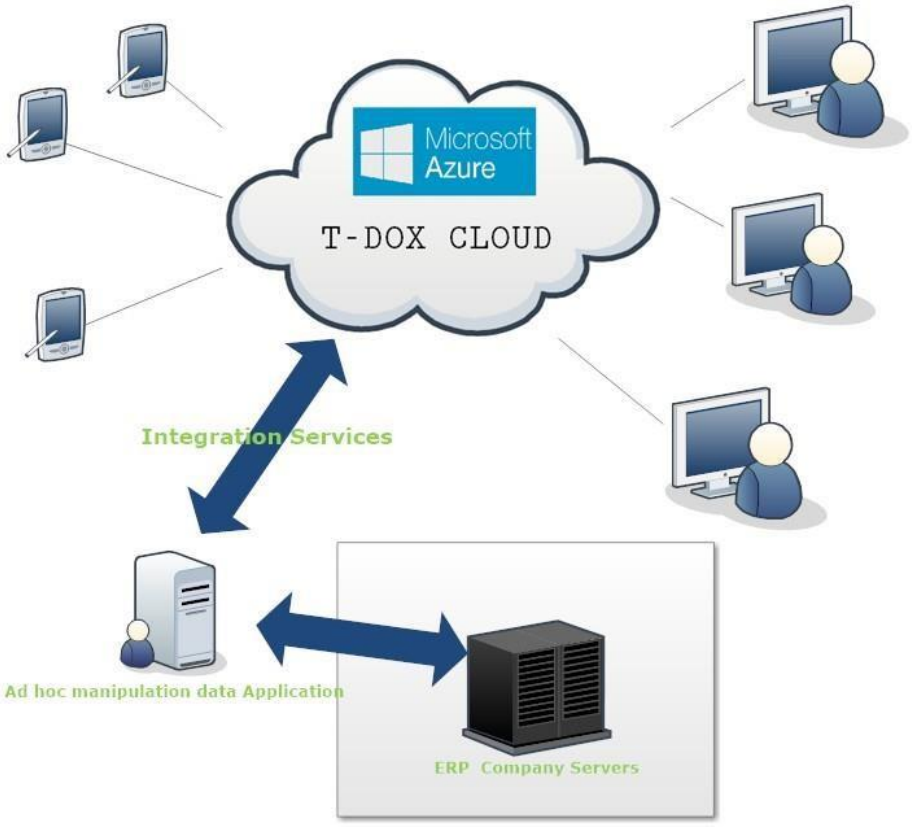
1. Introduction

The hereby document will guide you through the integration process and its procedure "TDox XCH" with the client's system.

To be precise, this document will guide you through:

- Programming XML applicative content for exchanged data;
- Individuating needed project, architectural and tecnological elements.

2. Integration Architecture

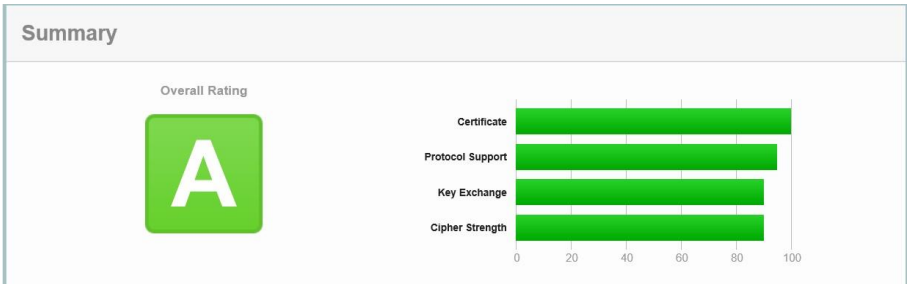


TDox is a CLOUD service, hosted by Microsoft Azure. All devices communicate with the cloud protocol HTTPS and are cifred in TSL 256 bit. The level of service guaranteed is 99.9%.

TDox Security Standards are valued on SSL-Labs, reaching level **A**:

WEBAPP: https://www.ssllabs.com/ssltest/analyze.html?d=webapp.mytdox.com

API: https://www.ssllabs.com/ssltest/analyze.html?d=exchangedata.mytdox.com



Through SOAP WEBAPI, it is possible to load the following data:

- Articles
- Categories
- Customers
- Events
- Lists
- User Lists
- Additional Data

While it is possible to download:

- Outcome of operator's operations
- PDF-A Files generated upon data collection
- Photos collected during operations

SOAP Services are at our client's disposal, they can be used to exchange data on SERVER to SERVER mode. Our Services won't accept more than 10 connections at the same time from the same customer, this is set in order to avoid its direct use on applications distributed with requests coming directly from clients. The XML format guarantees the correct data structure which can be gathered by TDox WSDL (More on this down below).
It is possible to view details about each connector at this address:

https://exchangedata.mytdox.com

On the page, you'll be able to see the current interface version and the previous ones.

3. Messaging definition

In order to grant full interoperativity, the application content of exchanged messages must follow XML syntax; hereby you can find the structure of exchanged messages.
Fields "DateTime" available both on request messages and reponse messages will have a structure composed as follows: <<Date>>T<<Time>>, where:

- <<date>>: AAAA-MM-GG
- <<time>>: HH:MM:SS

Example: "2007-12-31T17:25:50".

4. Web Services used by TDox

Authentication and Security:

Even communicating with TDox WEBAPI takes advantage of the HTTPS protocol cifred in SSL 256 bit.
All used services will be tagged <tokenWS>string</tokenWS>. TokenWS is released by TDox and must therefore be kept secret as it is a key for using our services and identifies univocally each active "company" in TDox.
It is also possible for a client to configure a limitation on domains that can request APIs relative to their data centre.

IMPORTANT: TDox uses precautions on accidental or malicious improper usage. It is not possible to request the same TOKEN or the same method more than 3 times in 30 seconds. Other than the request limit, each method has a size limit on exchanged data with TDox. As Follows, under the description of each single method, is reported a specific data size limit.

On each **RESULT** method you will be able to find the **TAG**:

```
<PropDDS>string</PropDDS>
```

In this given TAG is reported a particular code error which is returned in case too many requests have been made to the same token.

External Data Exchange to TDox:

Since version 2.0 it is available for each client\article\list a node called "Categories". Through that node, it is possible to assign categories to each object.

```
<EnableImportCategories>boolean</EnableImportCategories>
        <Categories>
          <string>string</string>
          <string>string</string>          </Categories>
```

Each Web service returns you the outcome of an operation for a single record, reporting for each record the following information:

- Code: Record Code
- ResultCode: Outcome Code {OK, KO or WA} (WA=warning)
- ResultString: Eventual explanation of the Outcome Code

ImportArticle

The client will invoke TDox service sending a SOAP (XML) message containing requested data.
Method Name: ImportArticle
Input: Please check the example below.
Expected Outcome: the service will import the input article, deleting it in case it should be identified as "To be deleted". Articles already existing in TDox will be updated if different compared to sent ones.
OUTPUT: After recording given data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportArticleAdditionalData

The client will invoke TDox Service sending a SOAP (XML) message containing requested data.
Method Name: ImportArticleAdditionalData
INPUT: Please check the example below.
Expected Outcome: the service will import additional data linked to the input article, deleting it in case this should be identified as "To be deleted". Already existing additional data will be updated if different compared to sent one.
OUTPUT: After recording given data, the system wil return either a message with a positive outcome (OK) or error codes.

ImportArticles

The client will invoke TDox service sending a SOAP (XML) message containing requested data.
Method Name: ImportArticles
Input: Please check the example below.
Expected Outcome: the service will import all input articles and delete all those idenitified as "To be Deleted". Articles already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.
Data Size Limit: It is possible to upload up to 100 records per request.

ImportCategories

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportCategories
Input: Please check example below.
Expected Outcome: The service will import input categories, deleting all those identified as "To be Deleted". Already existing categories will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.
Data Size Limit: It is possible to upload up to 100 records per request.

ImportCategory

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCategory
Input: Please check example below.
Expected Outcome: The service will import input category, deleting it in case the category should be identified as "To be Deleted". Categories already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCategoryAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing all requested data.
Method Name: ImportCategoryAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input category, deleting it in case additional data should be identified as "To be Deleted". Already existing additional data in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCenterAdditionalData

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCenterAdditionalData
Input: Please check example below.

Expected Outcome: The service will import additional data linked to input centre, deleting it in case the additional data should be identified as "To be Deleted". Additional Data already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCustomer

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCustomer
Input: Please check example below.
Expected Outcome: The service will import input customer, deleting it in case it should be identified as "To be Deleted". Customers already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCustomerAdditionalData

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCustomerAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input customer, deleting it in case additional data should be identified as "To be Deleted". Additional Data already existing in TDox will be updated if different compared to sent data.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCustomerCategories

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCustomerCategories
Input: Please check example below.
Expected Outcome: The service will import categories linked to input customer.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportCustomers

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportCustomers
Input: Please check example below.
Expected Outcome: The service will import all input customers, deleting those identified as "To be Deleted". Customers already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.
Data Size Limit: It is possible to upload up to 100 records per request.

ImportEvent

The client will invoke TDox service sending a SOAP (XML) message containing all requested data.
Method Name: ImportEvent
Input: Please check example below.
Expected Outcome: The service will import input event, deleting it in case it should be identified as "To be Deleted". Events already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportEvents

The Client will invoke TDox service by sending a SOAP message (XML) which will contain requested data.
Method Name: ImportEvents

Input: Please check example below.
This XML is structured with different components: Here there is the header:

```
<ModelTitle>string</ModelTitle>    --Process Title
<UserEmail>string</UserEmail>      --Assigned User
<IdForeignKeySystem>string</IdForeignKeySystem>   --External Event ID
<Start>dateTime</Start>       -- Start date and time
<End>dateTime</End>           -- End date and time
<AllDay>boolean</AllDay>      -- Flag for full day
<Note>string</Note>     -- Notes
```

On a detail section, it will be possible to pass data related to clients scheduled on a certain event:

```
<ImpEvtDetailCustomer>
            <TxValue>string</TxValue>             -- Client's code
            <IsFixedValue>boolean</IsFixedValue>  -- Not editable value
            <IdForeignKeySystem>string</IdForeignKeySystem>  -- External ID
            <CdName>string</CdName>               -- Client variable name
            <PrIndex>int</PrIndex>           -- Progressive Visualization index
</ImpEvtDetailCustomer>
```

On this detail section it will be possible to send to the Agenda, Serial Numbers scheduled for a certain event:

```
<ImpEvtDetailSN>
            <TxValue>string</TxValue>             -- Barcode
            <IsFixedValue>boolean</IsFixedValue>  -- Not modifiable value
            <IdForeignKeySystem>string</IdForeignKeySystem>  -- External ID
            <CdName>string</CdName>     -- Barcode variable name
            <PrIndex>int</PrIndex>       -- Progressive Visualization index
</ImpEvtDetailSN>
```

On this detail section it will be possible to send to the Agenda, products scheduled for a certain event:

```
<ImpEvtDetailProduct>
            <TxValue>string</TxValue>     -- Product's code
            <QtyValue>double</QtyValue>   --Quantity
<BarcodeValue>string</BarcodeValue>   -- Barcode
            <UMValue>string</UMValue>     --Unit of measure
            <DescriptionValue>string</DescriptionValue>  --Description
            <IsFixedValue>boolean</IsFixedValue> --Not modifiable value
            <IdForeignKeySystem>string</IdForeignKeySystem> --External ID
            <CdName>string</CdName> -- Product variable name
            <PrIndex>int</PrIndex>     -- Progressive Visualization index
</ImpEvtDetailProduct>
```

On this detail section it will be possible to send to the Agenda other elements included in a model and scheduled for a certain event:

```
<ImpEvtDetailGeneric>
            <TxValue>string</TxValue>             -- Variable value
            <IsFixedValue>boolean</IsFixedValue>  --Not modifiable value
            <IdForeignKeySystem>string</IdForeignKeySystem> -- External ID
```

```
            <CdName>string</CdName> -- Variable name
            <PrIndex>int</PrIndex>  -- Progressive Visualization index
</ImpEvtDetailGeneric>
```

Set the following parameter if you want to upload objects which could not belong to the current model and if you don't want TDox to control names coherence.

```
<flagSuppressCheckNameVariables>boolean</flagSuppressCheckNameVariables>
```

Near the XML closing line you can define filter fields:

```
<flagForceDelete>boolean</flagForceDelete>
```

Setting the previous parameter on "true" all events uploaded in TDox yet to be executed by the operators will be deleted before creating new events. Adding values to filters it will be possible to limit this deletion.
There can be 2 types of filters (TimeRange and ListObjects). Both typologies can be valued at the same time and will be in **AND** between each other.

```
      <filterRemover>
        <TimeRange_Filter>
          <Start>dateTime</Start>              --Start Date
          <End>dateTime</End>                  --End Date
          <Users>                               --Users list
            <string>string</string>
            <string>string</string>
          </Users>
          <Models>                              --Model list
            <string>string</string>
            <string>string</string>
          </Models>
        </TimeRange_Filter>
        <ListObjects_Filter>
          <List_Objects>
            <ObjectFilter xsi:nil="true" />
            <ObjectFilter xsi:nil="true" />
          </List_Objects>
        </ListObjects_Filter>
      </filterRemover>
```

ObjectFilter is composed of:

```
 string VarName                              --A model variable name
 ObjectFilterType VarType                   --Type of variable *
 List<string> TextValues                    --List of variable values **
 List<int> NumberValues                     --List of variable values **
```

```
 List<double> FloatValues          --List of variable values **
* Vartype can assume the following values:
- Barcode
- Product
- Customer
- Label
- TextBox
- List
- CategoryList

** TextValues, NumberValues, FloatValues are mutually exclusive. Only the
corresponding field must be filled, according to the contained data type on the
model (string, integerm, decimal).
```

Multiple assignation: Through each parameter you can find down below, it is possible to assign a specific event to more users simultaneously. The assignation according to categories assigned to specific users. By specifing a specific category, the request will create an event of each assigned user.

```
<categoryListToAssignate>
        <string>string</string>
        <string>string</string> </categoryListToAssignate>
```

In case of not-existing categories o in case of categories not associated to any user, the method will return **Error: Users not found**.

EXPECTED RESULT: The service will import all input events and delete all those identified as "To Be Deleted".
OUTPUT: The system, after recording data, will return an XML message with either a positive outcome (OK) or with messages.
Data Size Limit: It is possible to upload maximum 100 records per request

ImportList

The client will invoke TDox sending a SOAP message (XML) message which will contain requested data.
Method Name: ImportList

Input: Please check the example below.
Expected Outcome: The service will import input list , deleting it in case the list should be identified as "To Be Deleted". Already existing lists will have be updated if data available in TDox will be different compared to sent data.
Output: After recording all data, the system will return an XML message with either positive outcome (OK) or error codes.
Data Size Limit: It is possible to upload up to 100 records per request.

ImportListItemAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportListItemAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input list item, deleting it in case additional data is identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportLists

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportLists
Input: Please check example below.
Expected Outcome: The service will import input lists, deleting them in case lists are identified as "To be Deleted". Lists already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.
Data Size Limit : it is possible to upload up to 100 records per request.

ImportMultipleArticlesAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleArticlesAdditionalData
Input: Please check example below.
Expected Outcome: The service will import input lists, deleting them in case lists are identified as "To be Deleted". Lists already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportMultipleCategoriesAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleCategoriesAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input categories, deleting those identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportMultipleCustomersAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleCustomersAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input customers, deleting those identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportMultipleListItemsAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleListItemsAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input list elements, deleting those identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportMultipleOperatorsAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleOperatorsAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input operators, deleting data identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportMultipleOperatorAdditionalData

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportMultipleOperatorAdditionalData
Input: Please check example below.
Expected Outcome: The service will import additional data linked to input operator, deleting it, if identified as "To be Deleted". Additional data already existing in TDox will be updated if different compared to sent one.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

ImportUserList

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ImportUserList
Input: Please check example below.
Expected Outcome: The service will import input user list, deleting those identified as "To be Deleted". User lists already existing in TDox will be updated if different compared to sent ones.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

SendMessages

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: SendMessages
Input: Please check example below.
Expected Outcome: The service will import the list of messages for each specific customer, the customer will receive those messages as a notification on his device.
Output: After recording data, the system will return an XML message with either a positive outcome (OK) or error codes.

DeleteEvents

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: DeleteEvents
INPUT: Please check example below.
Expected Outcome: The service will delete non executed events using specified filters.
Output: After deleting data, thw system will return an XML message with either positive outcome (OK) or error codes.

External Data Exchange from TDox:

Since version 2.0 TDox uses methods to ask data to the cloud without using the PUSH method described down below. It will be up to the client to create those time procedures to interrogate TDox.
All methods accept and need pre-set filters in order to limit quantity of data to extract. The available paramenters are:

```
 <start>dateTime</start>          --Start Period
<end>dateTime</end>               --End Period
 <models>
  <long>long</long>               --Model ID to extract
   <long>long</long>              --Specific session to extract
 </models>
<instances>
   <guid>guid</guid>
   <guid>guid</guid>
 </instances>
```

The "Instances" TAG univocally identifies the execution of a certain model done by a certain user. It can be used when you want to download from TDox a PDF file for example, with a specific process.

The determination of identification marks (Instance ID, Model ID, etc.) or the determination of structure can be achieved in 2 different ways through GET methods described as follows:

GetInstancesExportDocuments

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetinstancesExportDocuments
Input: Please check example below.
Expected Outcome: The service will return the list of process instances and their related IDs assigned by TDox.
Data Size Limit: No limit.

GetModelRevisionInstancesInformation

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetModelRevisionInstancesInformation
Input: Please check example below.
Expected Outcome: The service will return, for each input instance, its related model ID and revision ID.
Data Size Limit: No limit.

GetModelRevisionsInformation

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetModelRevisionsInformation
Input: Please check example below.
Expected Outcome: The service will return the list of revisions for a certain Model and for each one of them, the ID which TDox assigned to them.
Data Size Limit: No limit.

GetModelsInformation

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetModelsInformation
Input: Please check example below.
Expected Outcome: The service will return the list of active\enabled models and for each one of them the ID which TDox assigned to them.
Data Size Limit: No limit.

GetOperators

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetOperators
Input: Please check example below.
Expected Outcome: The service will return the list of enabled users.
Data Size Limit: No limit.

GetUserListStructure

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: GetUserListStructure
Input: Please check example below.
Expected Outcome: The service will return the structure of a specific user list.
Data Size Limit: No limit.

Once gathered eventual IDs/Structures, it will be possible to request exportation or importation methods.

ExportDocuments

The client will invoke TDox service sending a SOAP message (XML) containing request data.
Method Name: ExportDocuments
Input: Please check example below.
Expected Outcome: The service will filter data and return all data collected during process instances.
Output: The System will return an XML message containing process data obtained by the fiven filter. Data will be wrapped by 2 precise tags, representing the process structure scheme and gathered data:

```
<XmlSchema>string</XmlSchema>
<XmlData>string</XmlData>
```

Data Dimension Limit:
ExportDocument Method needs a call that uses the maximum number of filters, expecially in those cases in which collected data quantities are quite high. This Method responds with an error if sent data is more than 1MB. In those cases is good to add new filters in order to secure data download. This method takes advantage of different filtering systems which work in AND mode between eachother.
Date Filter: If set, the method will only download documents contained in the selected date range.
Model Revisions: If set, the method will only download documents processed after the selected model revision.
Instances: If set, the method will only download documents related to the list of instances received by input
Operators: If set, this method will only download documents filled by the list of users received by input
ObjectFiltrer: If set, this complex object, allows to insert filters on text fields available on the filled document. (E.g. All documents which have the field OPERATOR filled as "RobertSmith").

ExportImages

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ExportImages
Input: Please check example below.
Expected Outcome: The service will filter data and return photos collected during processes.
Output: The system will return an XML message containing all photo files gathered by the requested filter.

ExportPDFDocuments

The client will invoke TDox service sending a SOAP message (XML) containing requested data.
Method Name: ExportPDFDocuments
Input: Please check example below.
Expected Outcome: The service will filter data, returning processe PDFs.
Output: The system will return an XML message containing all document files gathered by the requested filter.
Data Size Limit: 70 files or 2MB

5. Web Services exposed by the client (PUSH method)

The client will be able to expose one or more Web Services in order to be alerted of each document and in order to receive all documents filled by the operators. Once the document will be completed, TDox will call the related Web Service and will send its XML file containing all relevant information.

EXPOSED METHOD NAME: <**set by customer during set up>**
INPUT: String (Document of the filled document in XML format with the status of the document getting filled).
EXPECTED RESULT: TDox will invoke the service, providing you the following data in XML format:

- **Instance**: unique code (Guid) for the generated flux instance
- **DateStart**: flux start date
- **DateEnd**: flux end date
- **ModelId**: process type internal code
- **ModelName**: process title
- **Operator**: name of the operator who filled the flux and collected data
- **Status**: process (CLOSES = closed with success, ABORTED = aborted by user)

OUTPUT: The System will provide an XML message containing the status of any instance request.

TDox supports multiple exportation webmethods, it will be up to the customer to choose which one is more convenient . On the program side the function will have to be set, based on customer choice.

Webmethod with DataSet parameters: The webmethod will have 2 parameters as input and 1 as output

DataSet [Input]: DataSet is the input containing a list of DataTable associated to the exported Model. String [Input]: Unique description of the exported model

DataSet [Output]: is the returning DateSet, containing the DataTable which returns results from exported data.

*** At the end of the document you will find an example of method's signature and XML navigation.**

Webmethod with string parameters: This webmethod has 3 input parameters and 1 output parameter.

string [Input]: string containing an XML scheme related to the following

string [Input]: string containing DataSet XML returning all export values

string [Input]: Univocal description of the exported model

string [Output]: string containing DataSet XML returning exported data results.

*** At the end of this document you will find an example of method's signature and XML navigation**

Sent XML DataSet Structure

As described, TDox invokes the requested service by the invoker and sends an XML file containing all data collected by operators. This document represent a DataSet structured as follows:
HEADER TABLE: Table containing all previously described data, common between all processes (instance, Date Start, etc) + all not tabular data collected during the process.

Repeated data tabs: in each data collection process there can be 0 to N tables which can be filled by the operator. The XML file will contain from 0 to N additional tables to the header ones. In each additional table, will there always be an instance field which is an external key in the header tab.

In order to clarify everything we will show you here asimple example: Imagine that some of your users are using TDox to to collect purchase orders from their clients. A purchase order collection process is made of heading information (e.g. purchase date, client code, expected delivery date, order number) and a table where an operator can add multiple rows for each order (e.g. item number, quantity, price).
In this case, TDox will export to the invoker an XML file containing the HEADER TALE (instance, Date Start …. + Order Date, Client code …..) and an **additional table** containing data from rows related to each instance.

The name of the HEADER table will always be HEADER <process name>.

Names for the additional tables will always be customizable by the workflow designer, in this case then they will vary each time, even though they will be always followed by <process name>.

Hereby you can see the complete XML scheme. By following this example and supposing that the entire process has been called “Purchase order”, for each single order collected by users, TDox will export an CML structured as follows:

```
<NewDataSet>
   <HEADER_Purchase_Order>
       <Customer__ENTITY_REVISION>523d9684-d3a1-4d6d-96b2-91b6a2d0452f</Customer__ENTITY_REVISION>
              <Customer__DS_BUSINESS_NAME>Sebastian Wagner</Customer__DS_BUSINESS_NAME>
              <Customer__CD_ERP_CODE>IYD6896</Customer__CD_ERP_CODE>



       <Instance>2bdccb1d-0370-4d5b-462a-4ead6cde1dc2</Instance>
       <DateStart>2015-04-22T16:31:32.113+02:00</DateStart>
       <DateEnd>2015-04-22T16:31:50.113+02:00</DateEnd>
       <ModelId>20232</ModelId>
       <Status>CLOSED</Status>
       <ModelName>Purchase Order</ModelName>
       <Operator>prova@tdox.com</Operator>
       <Purchase_Date>2015-05-22T00:00:00.00+02:00</Purchase_Date>
  </HEADER_Purchase_Order>


  <DETAILS_Purchase_Order>
       <Instance>2bdccb1d-0370-4d5b-462a-4ead6cde1dc2</Instance>
       <PrIndex>0</PrIndex>
       <Product_Code>125689 </Product_Code>
       <Quantity>10</Quantity >
       <Price>1.56</Price>
  </DETAILS_Purchase_Order>
  <DETAILS_Purchase_Order>
       <Instance>2bdccb1d-0370-4d5b-462a-4ead6cde1dc2</Instance>
       <PrIndex>0</PrIndex>

       < Product_Code >025668 </Product_Code>
       <Quantity>15</Quantity>
       <Price>100.23</Price>
  </DETAILS_Purchase_Order>
<NewDataSet>
```

In each Details record the field Instance is available as external key to the HEADER key.

With each exportation, the XML file will always contain, as well as with relevant data, the structure scheme.

XML Response Dataset Structure

Webmethod XML Response must contain a DATASET containing just one table. This table must be called RESULTS. This table will have 2 string columns.

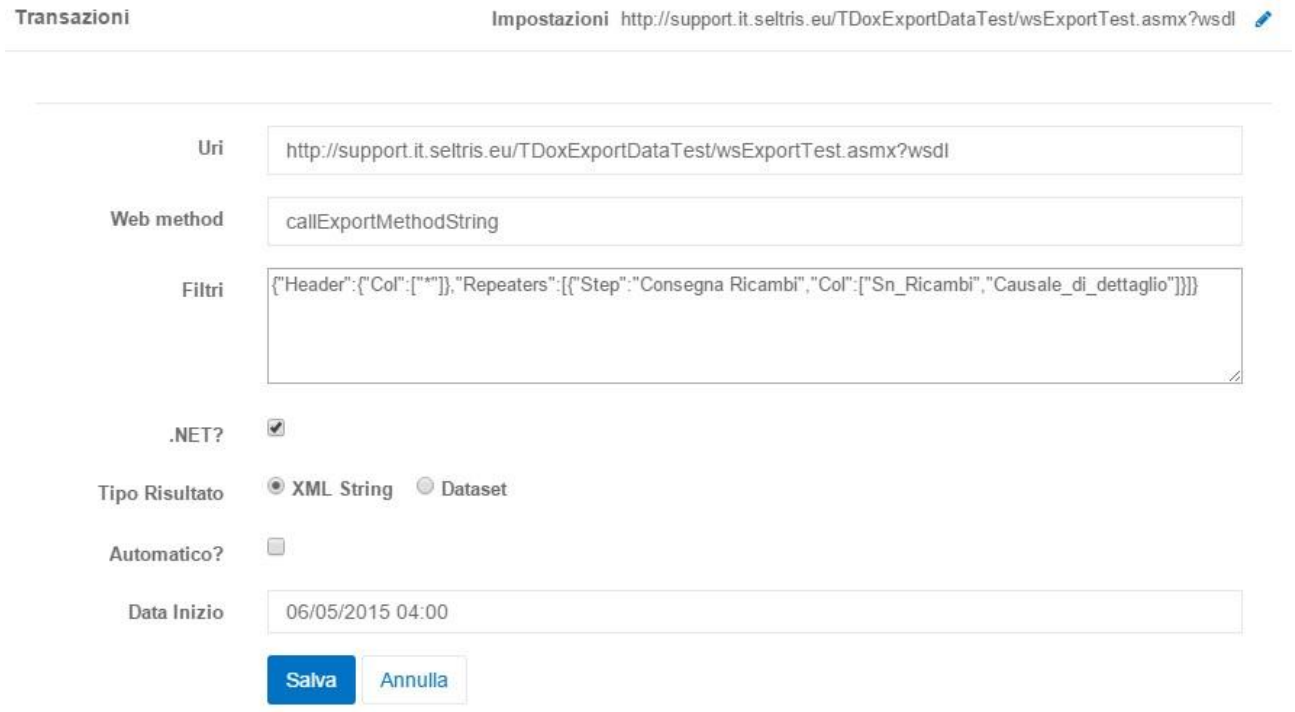
- Instance: Instance Code inserted in the datatable HEADER_ "..."
- RES: Result related to Instance (Accepted values: "OK", "KO")

6. Test and Setup

TDox Admins and System Integrators will be able to access to the EXHANGE DATA section on webapp from the title bar.

From the menu, it is possible to access pages for tests and setup of outgoing transactions.
To the right are displayed modules created with TDox. By selecting any of those modules, it is possible to setup requests to external webservices.

By pressing on the top right corner, it is possible to see settings for requests to external web services related to the selected module.



<u>The following options can be modified:</u>

URI: wsdl file path

Web Method: Web Service Requested name.

Filters: Name of objects that you want to export into the XML. Leaving this field empty, TDox will export all data available withing the module. Please consider using this filter to export only necessary information.

.NET? To be ticked only in case you are using a Microsoft.NET service

Result Type: You can setup type of data contained on the return object provided by the method: dataset or string.
Automatic?: To be ticked in case once finished testing, you would like to export automatically collected data.
Start Date: Date in which you would like to start exporting data automatically.

Once finished setting up , it is possible to save settings with the related button or to cancel changes.

On the table below, the program will show the last 10 effective transactions done by our operators. For each single one of them, it is possible to see an XML with all data related to the transaction by pressing on this icon  or to send a request to the external web service by clicking on  and waiting for the operation to end. In case everything should be successful, the program will return the XML generated upon external service. (XML Return).

Data filter field

The Data filter is a complex feature. Let's start saying that a process\document can be made of "flat" data such as header, footer and data contained within tables. In order to understand more efficiently this concept, please see **Sent XML dataset structure.**
By using data filter we can reduce to the minimum the XML size, which will then be exported by TDox to the external web service both header data and data contained by related tables.
The HEADER must be the first one to be available, within the header, all variables you want to export must be declared (with * you can export all columns). {"Header":{"Col":["*"]}}
The following example shows the exportation for all Header variables.

Other than "Header" there is another keyword: "Repeaters". In the "Repeaters" section you will be able to list each column you want to export from any Repeated STEP. Even in this case, with * you will be able to export the full list of variables.

{"Header":{"Col":["*"]},"Repeaters":[{"Step":"Expense","Col":["*"]}]}

In order to understand better how to set everything up, here is a practical example:

{"Header":{"Col":["*"]},"Repeaters":[{"Step":"Expense,"Col":["Date_1","Travelled_kilometers"]}]} This will export all header columns and columns from the repeated STEP "Expense". In this case, exported columns will be "Date_1" and "Travelled_kilometers".
{"Header":{"Col":["*"]},"Repeaters":[{"Step":"Expense,"Col":[" Date_1"," Travelled_kilometers "]}, {"Step":"Food_Expense,"Col":["Lunch","Dinner"]}]} This will export all header columns and columns from the repeated STEP "Expense" (Date_1,Travelled_Kilometers) and columns from the repeated STEP "Food_Expense" (Lunch, Dinner).
{"Header":{"Col":["First_and_last_name ", "Registration_Number"]}, "Repeaters":[{"Step":"Expense,"Col":["*"]}]} This will export "First_and_last_name " and "Registration_Number" Header tabs as well as with ALL columns from the repeated step "Expense".

Example of XML scheme sent by TDox to an external webservice

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
 <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">

<xs:complexType>

    <xs:choice minOccurs="0" maxOccurs="unbounded">
     <xs:element name="HEADER_Example">
      <xs:complexType>
       <xs:sequence>
         <xs:element name="Client__ENTITY_REVISION" msdata:DataType="System.Guid, mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" type="xs:string" minOccurs="0" />
         <xs:element name="Client__IS_PERSON" type="xs:boolean" minOccurs="0" />
         <xs:element name="Client__DS_BUSINESS_NAME" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_FIRST_NAME" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_LAST_NAME" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_FULL_NAME" type="xs:string" minOccurs="0" />
         <xs:element name="Client__CD_ERP_CODE" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_PHONE1" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_FAX1" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_MOBILE1" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_EMAIL" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_WEB_SITE" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_TAX_CODE" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_VAT_CODE" type="xs:string" minOccurs="0" />
         <xs:element name="Client__CD_STATUS" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_ADDRESS" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_ZIP" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_CITY" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_PROVINCE" type="xs:string" minOccurs="0" />
         <xs:element name="Client__TX_COUNTRY" type="xs:string" minOccurs="0" />
         <xs:element name="Notes" type="xs:string" minOccurs="0" />
         <xs:element name="Gps" type="xs:string" minOccurs="0" />
         <xs:element name="Gps__LATITUDE" type="xs:string" minOccurs="0" />
         <xs:element name="Gps__LONGITUDE" type="xs:string" minOccurs="0" />
         <xs:element name="Gps__PROVIDER" type="xs:string" minOccurs="0" />
         <xs:element name="Gps__TIMESTAMP" type="xs:dateTime" minOccurs="0" />
         <xs:element name="STEP_1__COUNT" type="xs:double" minOccurs="0" />
         <xs:element name="Instance" type="xs:string" minOccurs="0" />
         <xs:element name="DateStart" type="xs:dateTime" minOccurs="0" />
         <xs:element name="DateEnd" type="xs:dateTime" minOccurs="0" />
         <xs:element name="ModelId" type="xs:int" minOccurs="0" />
         <xs:element name="ModelName" type="xs:string" minOccurs="0" />
         <xs:element name="Status" type="xs:string" minOccurs="0" />
         <xs:element name="Operator" type="xs:string" minOccurs="0" />
         <xs:element name="ID_FKS" type="xs:string" minOccurs="0" />
```

```
        <xs:element name="Client__ENTITY_REVISION__ID_FKS" type="xs:string" minOccurs="0" />
        <xs:element name="Client__ENTITY_REVISION__PLANNED" type="xs:string" minOccurs="0" />
        <xs:element name="Note__ID_FKS" type="xs:string" minOccurs="0" />
        <xs:element name="Note__PLANNED" type="xs:string" minOccurs="0" />
</xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="DETAILS_Example">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="Serial_Number" type="xs:string" minOccurs="0" />
        <xs:element name="Notes" type="xs:string" minOccurs="0" />
        <xs:element name="Instance" type="xs:string" minOccurs="0" />
        <xs:element name="PrIndex" type="xs:int" minOccurs="0" />
        <xs:element name="Serial_Number__ID_FKS" type="xs:string" minOccurs="0" />
        <xs:element name="Serial_Number__PLANNED" type="xs:string" minOccurs="0" />
        <xs:element name="Notes__ID_FKS" type="xs:string" minOccurs="0" />
        <xs:element name="Notes__PLANNED" type="xs:string" minOccurs="0" />
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:choice>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

Example of XML sent by TDox to an external webservice

```
<NewDataSet>
 <HEADER_Example>
  <Client__ENTITY_REVISION>523d9684-d3a1-4d6d-96b2-91b6a2d0452f</Client__ENTITY_REVISION>
  <Client__DS_BUSINESS_NAME>Sebastian Wagner</Client__DS_BUSINESS_NAME>
  <Client__CD_ERP_CODE>IYD6896</Client__CD_ERP_CODE>
  <Gps>Via Giuseppe Mazzini, 30,
46043 Castiglione delle Stiviere MN,
</Gps>
  <Gps__LATITUDE>45,3840157</Gps__LATITUDE>
  <Gps__LONGITUDE>10,4924296</Gps__LONGITUDE>
  <Gps__PROVIDER>network</Gps__PROVIDER>
  <Gps__TIMESTAMP>2015-04-22T16:31:42.29+02:00</Gps__TIMESTAMP>
  <STEP_1__COUNT>1</STEP_1__COUNT>
  <Instance>2bdccb1d-0370-4d5b-462a-4ead6cde1dc2</Instance>
  <DateStart>2015-04-22T16:31:32.113+02:00</DateStart>
  <DateEnd>2015-04-22T16:31:50.113+02:00</DateEnd>
  <ModelId>20232</ModelId>
  <ModelName>Modello di Esempio</ModelName>
  <Status>CLOSED</Status>
```

<Operator>info@mytdox.com</Operator>
</HEADER_Example>
<HEADER_Example>
<Client__ENTITY_REVISION>4d88183a-ab9f-4c91-aaea-830c51188a11</Client__ENTITY_REVISION>
<Client__DS_BUSINESS_NAME>Cooper Bryan</Client__DS_BUSINESS_NAME>
<Client__CD_ERP_CODE>CKS1576</Client__CD_ERP_CODE>
<STEP_1__COUNT>0</STEP_1__COUNT>
<Instance>b21cdfda-57f3-4d66-6965-511fc37b3743</Instance>
<DateStart>2015-04-22T16:32:45.067+02:00</DateStart>
<DateEnd>2015-04-22T16:32:55.01+02:00</DateEnd>
<ModelId>20232</ModelId>
<ModelName>Modello di Esempio</ModelName>
<Operator>info@mytdox.com</Operator>

<Status>CLOSED</Status>
</HEADER_Example>
<HEADER_Example>
<Client__ENTITY_REVISION>d6dcdf41-b873-4aee-98b5-a988ff479df7</Client__ENTITY_REVISION>
<Client__DS_BUSINESS_NAME>Jocelyn Guy</Client__DS_BUSINESS_NAME>
<Client__CD_ERP_CODE>EXK9296</Client__CD_ERP_CODE>
<STEP_1__COUNT>1</STEP_1__COUNT>
<Instance>2fc5d6aa-0d02-4700-4e01-d3a2b82876d9</Instance>
<DateStart>2015-04-22T16:32:15.817+02:00</DateStart>
<DateEnd>2015-04-22T16:32:38.6+02:00</DateEnd>
<ModelId>20232</ModelId>
<ModelName>Example</ModelName>
<Operator>info@mytdox.com</Operator>
</HEADER_Example>
<HEADER_Example>
<Client__ENTITY_REVISION>d6dcdf41-b873-4aee-98b5-a988ff479df7</Client__ENTITY_REVISION>
<Client__DS_BUSINESS_NAME>Jocelyn Guy</Client__DS_BUSINESS_NAME>
<Client__CD_ERP_CODE>EXK9296</Client__CD_ERP_CODE>
<Notes>ciao
</Notes>
<Gps>Via Fratelli Kennedy, 30,
46043 Castiglione delle Stiviere MN,
</Gps>
<Gps__LATITUDE>45,3841088</Gps__LATITUDE>
<Gps__LONGITUDE>10,4953953</Gps__LONGITUDE>
<Gps__PROVIDER>network</Gps__PROVIDER>
<Gps__TIMESTAMP>2015-04-22T16:30:49.553+02:00</Gps__TIMESTAMP>
<STEP_1__COUNT>2</STEP_1__COUNT>
<Instance>e459f818-575d-484c-7bac-de8f11bf7044</Instance>
<DateStart>2015-04-22T16:30:04.573+02:00</DateStart>
<DateEnd>2015-04-22T16:31:24.56+02:00</DateEnd>
<ModelId>20232</ModelId>
<ModelName>Modello di Esempio</ModelName>
<Operator>info@seltris.it</Operator>

```
  <Status>CLOSED</Status>
 </HEADER_Example>
 <Example_DETAILS>
  <Serial_Number>gghhh</Serial_Number>
  <Instance>2bdccb1d-0370-4d5b-462a-4ead6cde1dc2</Instance>
  <PrIndex>0</PrIndex>
 </Example_DETAILS>
 <Example_DETAILS>
  <Serial_Number>rghu</Serial_Number>
  <Instance>2fc5d6aa-0d02-4700-4e01-d3a2b82876d9</Instance>
  <PrIndex>0</PrIndex>
 </ Example_DETAILS >
 < Example_DETAILS >
  <Serial_Number>fghh</Serial_Number>
  <Notes>ghhh</Notes>
  <Instance>e459f818-575d-484c-7bac-de8f11bf7044</Instance>    <PrIndex>0</PrIndex>
 </ Example_DETAILS >
 < Example_DETAILS >
  <Serial_Number>fhvff</Serial_Number>
  <Notes>ghhh</Notes>
  <Instance>e459f818-575d-484c-7bac-de8f11bf7044</Instance>
  <PrIndex>1</PrIndex>
 </ Example_DETAILS >
</NewDataSet>
```

XML RESULTS SCHEME:

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
 <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
  <xs:complexType>
   <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="RESULTS">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Instance" type="xs:string" minOccurs="0" />
       <xs:element name="RES" type="xs:string" minOccurs="0" />
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:choice>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

XML Results Example:

```
<NewDataSet>
 <RESULTS>
  <Instance>5d8922c2-7d33-4da6-5a98-2e7c04f028b3</Instance>
  <RES>OK</RES>
 </RESULTS>
 <RESULTS>
  <Instance>27858136-8528-476b-56fc-696601a58239</Instance>
  <RES>OK</RES>
 </RESULTS>
 <RESULTS>
  <Instance>89a4d4e0-c3d2-4440-50a3-71b68c2f43cb</Instance>
  <RES>OK</RES>
 </RESULTS>
 <RESULTS>
  <Instance>c22d951e-a1b9-4394-7ed5-eb64d3ceb232</Instance>
  <RES>OK</RES>
 </RESULTS>
 <RESULTS>
  <Instance>d4fc0a73-8b56-4b42-68cf-f91ccd8e89d8</Instance>
  <RES>OK</RES>
 </RESULTS>
</NewDataSet>
```

Signature Example of an external method requested by TDox:

```
using System; using

System.Collections.Generic; using

System.Data; using System.IO;

using System.Linq; using

System.Net; using

System.Net.Sockets;

using System.Text; using

System.Web; using

System.Web.Services;


namespace WebServiceExportTest
{
   /// <summary>
   /// Summary description for WSExportTest
```

```
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.ComponentModel.ToolboxItem(false)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
// [System.Web.Script.Services.ScriptService]
```

public class

WSExportTest : System.Web.Services.WebService

```
{
    [WebMethod]
public DataSet callExportMethod(DataSet ds,
string modelTitle)
    {
        // Dataset return
        DataSet dsRet = new DataSet();

        // Unique Dataset Table returned
        //   RESULTS   Table   name
// fields:
        //     Instance (string type)
        //     RES (string type ["OK","KO"])
        DataTable dtRet = new DataTable("RESULTS");
dtRet.Columns.Add(new DataColumn("Instance", typeof(String)));
dtRet.Columns.Add(new DataColumn("RES", typeof(String)));
        dsRet.Tables.Add(dtRet);

        // Retrieving sent table name
        // This is how it will look like:
        // HEADER_ + ... Model name

        List<DataTable>
dtHeaders
=
new
List<DataTable>();
foreach(DataTable dtIn in ds.Tables )
        {
          string
tableName
=
dtIn.TableName.ToUpper();
if(tableName.StartsWith("HEADER_"))
          {
            dtHeaders.Add(dtIn);
          }
        }
```

```
        // Preparing Return DataTable
foreach(DataTable dtIn in dtHeaders)
        {
           if (dtIn != null && dtIn.Rows != null && dtIn.Rows.Count > 0)
           {
              int id = 0;
              foreach (DataRow dr in dtIn.Rows)
              {
                 DataRow drOut = dtRet.NewRow();
drOut["Instance"] = dr["Instance"];
                 id++;
if (( id % 2)==0)
                 {
                    drOut["RES"] = "OK";
                 }
else                 {
drOut["RES"] =
"KO"; }



                 drOut["RES"]                 =                 "OK";
dtRet.Rows.Add(drOut);
              }
           }
        }
        return dsRet;
     }
     [WebMethod]
 public string callExportMethodString(string strdsSchema, string strds,string modelTitle)
     {
        // Method similiar to the previous one with additional string parameter
        //Strings      will      be      in      XML:
//strdsSchema = This will contain the XML
scheme
        //  strds = this will contain Dataset XML.
        //  Returned value will be the XML  gathered from Returned Dataset serilized structure.
```

```
// See example in webmethod callExportMethod


        DataSet ds = getDataSetFromString(strds);
DataSet dsRet = callExportMethod(ds, modelTitle);
return dsRet.GetXml();
      }


      private DataSet getDataSetFromString(string xmlString)
      {
        DataSet dsRes = new DataSet();           byte[] byteArray
= Encoding.UTF8.GetBytes(xmlString);
        MemoryStream stream = new MemoryStream(byteArray);
        StreamReader reader = new StreamReader(stream);


        dsRes.ReadXml(reader);
        return dsRes;
      }


   }
}
```