Trimble

Trimble Buildings
GC/CM Division

Vico Office Web Services – API Documentation

March

2017

This document includes all of the necessary information about Vico Office Web Services (VOWS) module.

Meridian Systems   PLANCAL   SketchUp   TEKLA   VICO SOFTWARE   winest

Trimble Buildings
GC/CM Division

Trimble.

Vico Office Web Services Documentation

Introduction

Vico Office Web Services provides a platform for accessing Vico Project Server (VPS) data. With VOWS, you have the means to:

- Expose VPS data from any host within your network through a unified interface to external systems across the Internet in a safe and secure manner
- Integrate Vico Office with other systems through a standardized platform
- Import or export files on a scheduled basis

Vico Office Web Services allow users to:

- Access VPS data in XML format through a simple SOAP data query service
- Get JSON format data through a REST interface
- Perform export or import operations on a predefined schedule from or to files having custom data formats

Getting Started

To facilitate the integration with the client application, both SOAP and REST type of web services are available. Both present the same information in either a Data Exchange XML or a JSON format. Working with SOAP or REST services requires an understanding of the way in which data of these services are exchanged between the client and the server application.

The Data Exchange XML format has a distinct XML type for each exposed CIM type. The format is defined in a set of XSD files. The XML service exchanges information by means of three XML types, the request document, the response document and the transmit document, that embed all the information needed to complete a call. Since the XML service is simplified to just two operations, export and import, understanding the structure of these documents is of paramount importance.

The JSON format provides a generic structure for all exposed CIM data by means of views, records and properties. This simplifies the development of user interfaces.

Clients using Microsoft .Net can generate an object-oriented client layer with strongly typed service proxies and document types for the XML service or can manually create strongly typed entity classes for the REST service data. Both service types have full support in Microsoft .Net.

TABLE OF CONTENTS

Introduction..1

Getting Started 1

SOAP Services & the Data Exchange XML Format 3

REST Service & JSON Data Structure 9

Vico Connector Service & Configurator..15

Data Structures..29

Using VOWS..49

SOAP Services & the Data Exchange XML Format

Two distinct services, a buffered and a streamed SOAP service, are the gateway to VPS information. Both have a simple interface consisting of just two methods:

Buffered service

- *ExportData* for requesting VPS data – it receives two strings parameters:
  - A string of a serialized *System.Guid* type acting as session identifier
  - A string containing an XML of a *RequestDocument* type

  It responds with a string containing an XML of a *ResponseDocument* type

- *ImportData* for creating or updating VPS data – it receives two strings parameters:
  - A string of a serialized *System.Guid* type acting as session identifier
  - A string containing an XML of a *TransmitDocument* type

  It responds with a string containing an XML of a *ResponseDocument* type

Streamed service

- *ExportData* for requesting VPS data – it receives a *RequestData* object that has two members:
  - *SID*: is a string of a serialized *System.Guid* type acting as session identifier
  - *XMLData*: is a string containing an XML of a *RequestDocument* type

  It responds with a *StreamData* object that has four members:

  - *SID*: is a string of a serialized *System.Guid* type acting as session identifier
  - *Length*: is the length of the stream
  - *IsZipped*: is a boolean specifying if the stream is a zip or a text file
  - *Stream*: is a *System.IO.Stream* to a data structure comprising an XML of a *ResponseDocument*
- *ImportData* for creating or updating data – it receives a *StreamData* object (see description above), where the *Stream* member is a stream to a data structure comprising an XML of a *TransmitDocument* and responds with a *StreamData* object, where the *Stream* member is a stream to a data structure comprising an XML of a *ResponseDocument*

All the types mentioned above are defined in the *RequestDocument.xsd*, *ResponseDocument.xsd* and *TransmitDocument.xsd* schema definition files. These can be seen as message envelopes containing a single *vowsdoc* element. For a comprehensive overview of these types, see the Data Exchange XML Format section.

For each document t*he Header* element contains generic information about server, project or data formatting while the *Body* element holds either information about the requested data or data itself.

The following Vico Office resources are exposed through VOWS:

- *CostPlan*: exposes all cost related data of Vico Office that is available in Vico Office's Cost Planner view
- *TakeOffSystem*: exposes takeoff related information of TOIs with TOQs, TOI related cad elements with each cad element's locations, and TOI related locations with each locations' TOI cad elements and by location TOQ values
- *LBS*: exposes location and location system related information
- *SchedulingSystem*: exposes scheduling related information of sum, schedule, location, detail and detail-location tasks with latest progress entry information for location tasks (task actuals)
- *CADModelSystem*: exposes cad model related information of a project as cad models, model versions, element and derived elements without 3D information for elements
- *TagSystem*: exposes tag related information of tag categories, tags and tag values
- *WorkPackageSystem*: exposes basic work package information
- *ConstructabilitySystem*: exposes constructability information including screenshots and images
- *VPS*: exposes available projects of one or all reachable VPS hosts in the LAN
- Data Exchange XML Format

The Data Exchange XML format has a distinct XML type for each exposed CIM type. The XML service exchanges information by means of three XML types/documents: *RequestDocumentType*, *ResponseDocumentType* and *TransmitDocumentType*, which embed all the information needed to complete either an export or an import call. These are defined in the *Common\RequestDocument.xsd*, *Common\ResponseDocument.xsd* and *Common\TransmitDocument.xsd* schema definition files. All XML Schema Definition files are within the folder *\Schemas\DataExchange\v1.0\*.

Each document has a *vowsdoc* element that has three attributes, all having fixed values:

- *messageSystem* fixed to *VOWSXML*
- *messageType* fixed to one of the followings: *RequestDocument, ResponseDocument, or TransmitDocument*, depending on the actual document type
- version fixed to *1.0*.

Each document has a single *vowsdoc* element that has a single *header* and a single *body* or *content* element. The *content* element holds the exposed CIM data structures.

The XML data structures of the *content* element follow the build-up logic of their CIM counterparts. The XML counterpart of each exposed CIM type have a *loid* attribute based on the CIM object's logical database identifier (loid for short); this is used to uniquely identify an XML element within a response or transmit document. XML types that represent items of a tree data structure in CIM have a second loid attribute called *ploid* (from parent loid); the XML element of the parent should be present before any children reference them. The *key* and *logicalType* attributes are optional; they are present only if their CIM counterpart creation is possible. The *key* attribute uniquely identifies the item within its own collection while the *logicalType* attribute holds complementary information about the CIM type. For a detailed description of the resources see Data Structures.

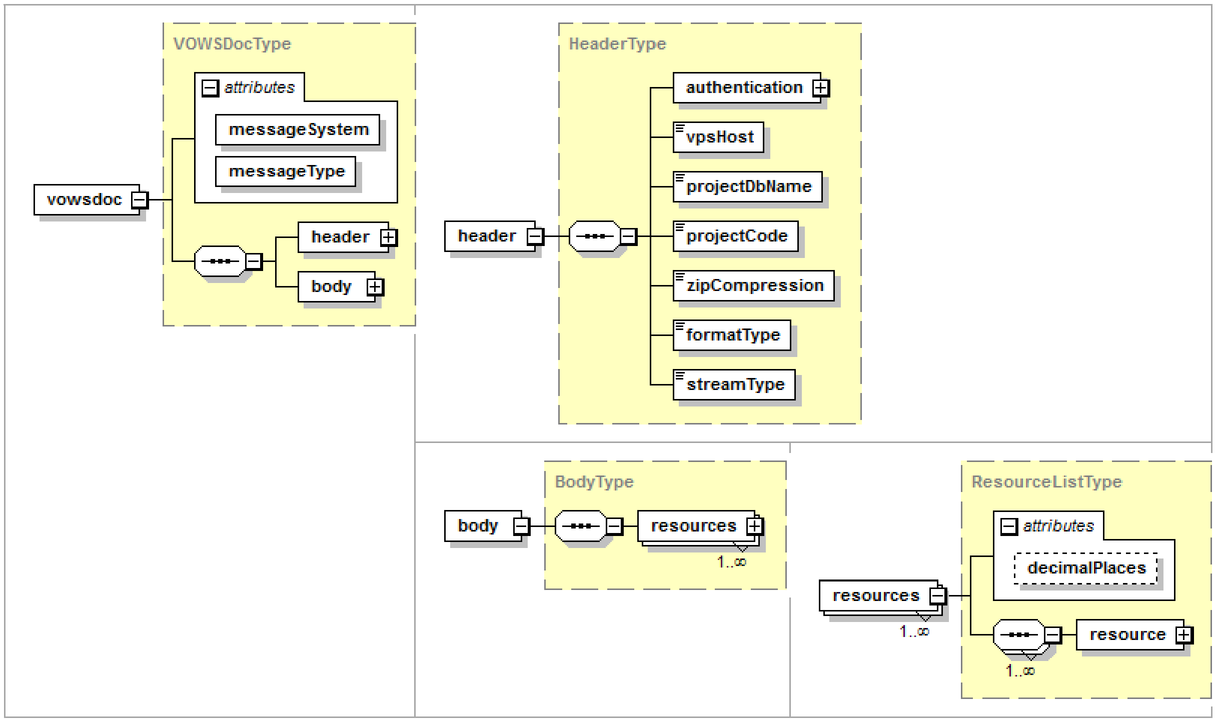
Request Document

A request document is used to configure what data the export service should provide in the response document.

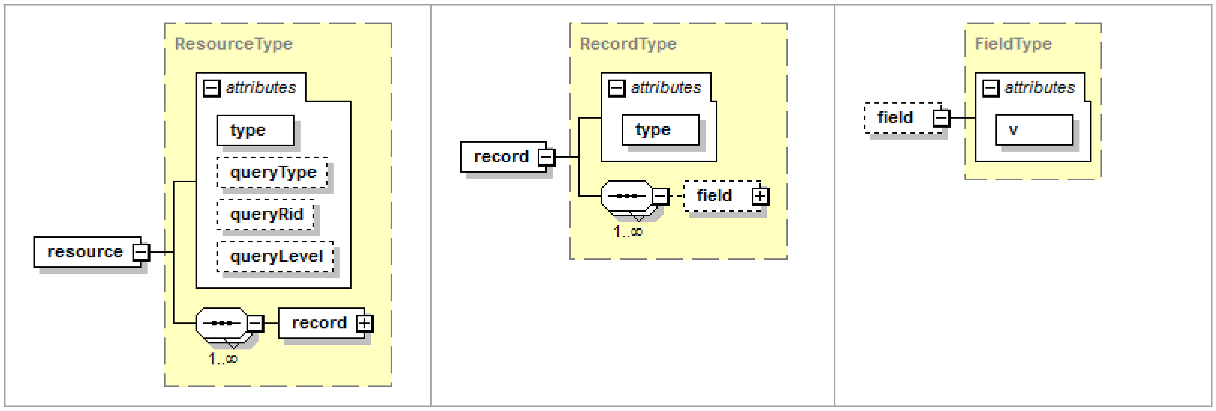
The *header* element contains the followings:

- Authentication information (reserved for future use)
- Project specific information as VPS host name, project code or project database name
- Format type of the response data, as *TypedXML* or *JSON*
- Stream type as *Memory* or *File*, specifying where the serialization of the response should take place on the server: in *Memory*, being faster but requiring more memory for large exports or in *File*, being slower but requiring less memory. The default is *Memory* type stream
- *zipCompression*, which is used only by the streamed XML service, sets whether the response should be zip compressed

The *body* element holds a list of requested resources in a *ResourceListType* xml element. The optional *decimalPlaces* attribute, common to all resources, defaults to 12 and sets the precision of the exported decimal values of the resources.



A resource is described by the *ResourceType* xml type. Its *type* attribute sets one of the exported cost management resources, as follows: *CADModelSystem*, *CostPlan*, *LBS, SchedulingSystem, TagSystem*, *TakeOffSystem, WorkPackageSystem, ConstructabilitySystem, VPS.*



QueryType and *QueryRid* together specify the association of the returned records with the given record id. As it is detailed in the REST Service & JSON Data Structure section, a record wraps a CIM object and it has a record ID that uniquely identifies it within a resource. A resource is made of a tree structure of records. The first record of the tree, also called the root record, always wraps the project CIM type and has 0 as the record id.

QueryType can be one of the followings: *GetChildren*, *GetChildrenRecursively*, *GetItem*, *GetParents*, their name being descriptive enough regarding the represented association. Both attributes are optional, their default value is *GetItem* for *QueryType* and *0* for *QueryRid. QueryLevel* is reserved for future use.

Note: *TypedXML* format requires *QueryRid* to be *0* and *QueryType* to be *GetChildrenRecursively*.

A record is described by the *RecordType* XML type. Its *type* attribute sets one of the record types used in record trees, such as: *CADModelSystem_CADModel*, *CostPlan_Component*,*TakeOffSystem_TOI*, etc.. For a complete list of record types consult *RecordTypeEnum* from *ResourceDefinitions.xsd*.

A field is described by the *FieldType* xml type. Its *v* attribute sets a property of the record, like: *CADModel_modelID*, *Component_code*, *TOI_name*. For a complete list of field types, consult *FieldTypeEnum* from *ResourceDefinitions.xsd*.

Since each field is meaningful only within a specific record and a record is meaningful only within a specific resource, the request document has to respect this convention in order to be accepted by the data query service. The naming convention used for record types and field types helps to identify which record belongs to which resource and which field belongs to which record. The name of a record type and that of a field type is made of two parts split by underscore '_'. For a record, the first part specifies the resource it belongs to and the second the actual record type: *CostPlan_Component*. Similarly, for a field, the first part specifies the record it belongs to and the second the actual field type: *Component_code*.

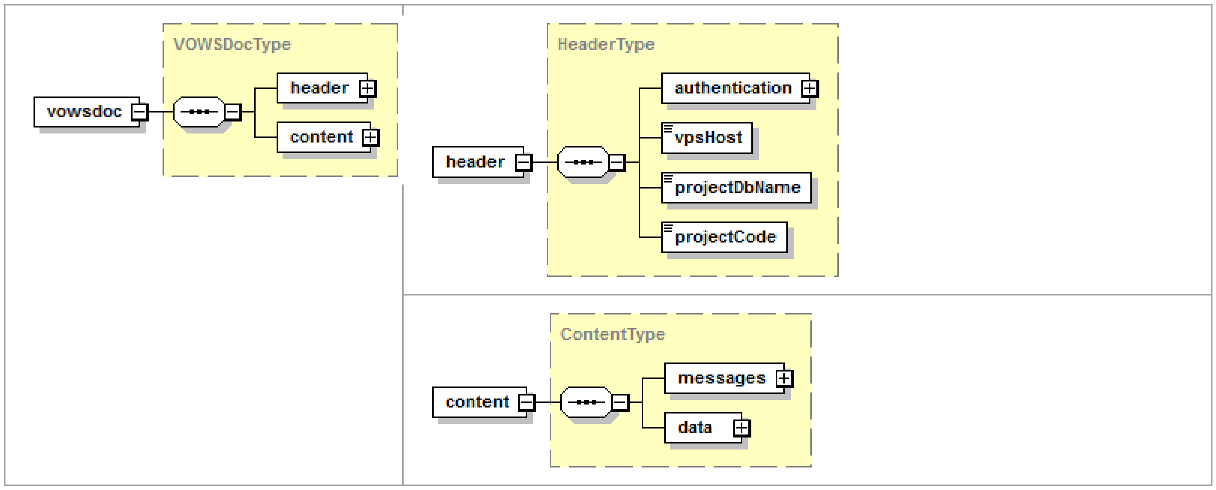
The data query service validates the request document for both its compliance with *RequestDocument.xsd* and for valid buildup logic.

Response Document

A response document is returned for both export and import service calls. It contains information about the accessed VPS host and project, messages generated during the call and the CIM data for an export call.

The *header* element contains authentication information (reserved for future use) and project specific information, such as VPS host name, project code and database name.

The *content* element contains the requested data and/or messages generated during the data request.



During a service call, several expected and unexpected events can happen that might hold valuable information for the caller. The data service's notification policy demands that all events to be notified to the caller unless the service itself crashes. In such case, the service connector's log should be checked. For further information, see Vico Connector Configurator.

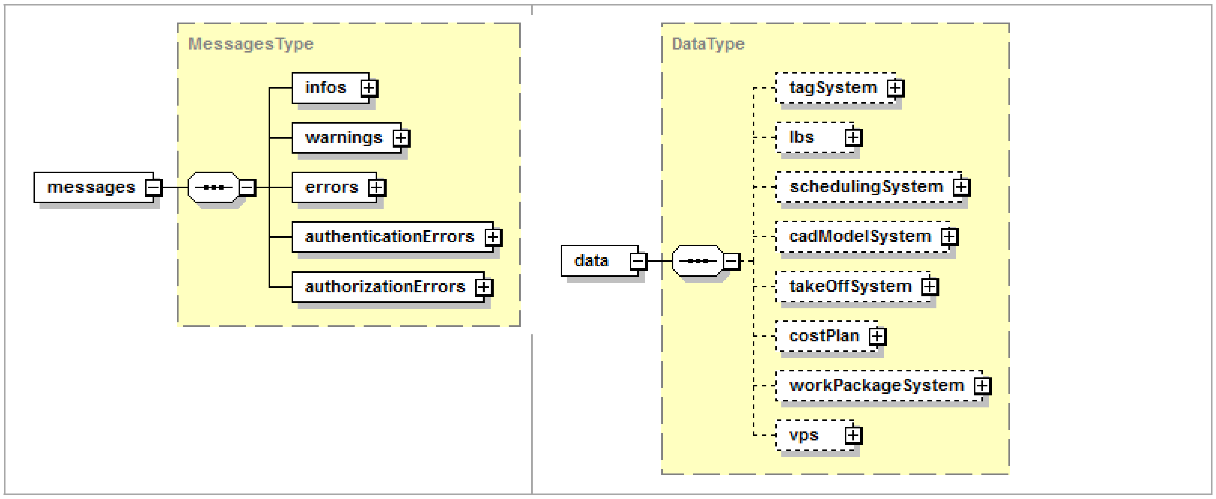
Messages generated during a call are split into three types:

- Info messages notifying general actions (e.g. import was finished successfully)
- Warning messages notifying about flaws that didn't stop the current action to be completed but can point to possible inconsistencies
- Error messages notifying about major problems that prevented the current action to be completed.

Requested data is returned in one of the resource specific elements:

- *tagSystem – Tag_System.xsd*
- *lbs – LBS.xsd*
- *schedulingSystem – Scheduling_System.xsd*
- *cadModelSystem – CAD_Model_System.xsd*
- *takeOffSystem – TakeOff_System.xsd*
- *costPlan – Cost_Plan.xsd*
- *workPackageSystem – Work_Package_System.xsd*
- *constructabilitySystem – Constructability_System.xsd*
- *vps - VPS_Info.xsd*

Schema Definition files are within the folder \Program Files\Vico Software\VOWS\Vico Office Services*Schemas**DataExchange**v1.0*\Cost_Management\.



A response to an import action has the *data* element empty as it only contains messages about the import action.

Transmit Document

The import method of the service expects a transmit document XML and responds with a response document XML. A transmit document can be used to create new items or to update existing ones. Only those CIM types marked as *Can Create*, or *Can Update* in the Data Structures section can be created and/or updated, the others are read-only. The service validates the transmit document before any import action is started, if read-only fields are found a response document with error notifications is returned as response.

Updating existing items

When using an XML element to update a CIM object counterpart its *loid* attribute must be valid within the specified project. The update action uses the XML element's *loid* attribute to look up for the specific CIM object in the project then performs the updating of the given fields. The only condition a *loid* attribute must met is to be valid within the given project; otherwise, a warning message notifies that the update could not be performed. A *loid* is not valid if it does not belong to the specified project or it was deleted in the meantime.

Creating new items

When using an XML element to create a CIM object counterpart, its *loid* attribute has to start with the plus '**+**' character. The rest of the *loid* can be any custom string that uniquely identifies the XML element within the import document. This custom, user-defined *loid* string is used solely to uniquely identify the XML element during the import. The CIM object counterpart created by virtue of it will get its own valid *loid* value when the object is actually created.

In both cases the whole data structure from the root element down to the updateable or creatable XML resource elements should be present in the transmit document. This convention is checked during the validation phase of the import process.

The transmit document buildup is similar to that of the response document even though not all XML resource types are updateable and/or creatable. Importable data could be present in the following resource elements:

- *tagSystem – Tag_System.xsd*
- *lbs – LBS.xsd*
- *schedulingSystem – Scheduling_System.xsd*
- *takeOffSystem – TakeOff_System.xsd*
- *costPlan – Cost_Plan.xsd*
- *workPackageSystem – Work_Package_System.xsd*
- *constructabilitySystem – Constructability_System.xsd*

Schema Definition files are within the folder \Program Files\Vico Software\VOWS\Vico Office Services*Schemas**DataExchange**v1.0*\Cost_Management\.

REST Service & JSON Data Structure

The REST service exposes data in JSON format about hosts available on the network, projects on a host and data from resources.

The data structure of resources is common for all and is composed of three types:

- *View:* Denotes the requested resource.
- *Record:* Wraps a CIM type.
- *Property:* Exposes a single attribute of the type.

The association between them is that a view has one to many records (the root record is always the project record) and a record has zero to many properties.

There are two types of properties: static and dynamic. Static properties are directly related to the record (e.g. Component – code or TOQ – name) while dynamic properties are created for a record by means of an association of the record's object with another object: e.g. Component *by* location properties (as quantity, price, total price); or takeoff quantity *by* location properties (as its value by location property). These '*by something*' properties are grouped together within a *Group*. Beside the dynamic properties, a group has a type and a caption property, with the caption showing the *loid* (Logical DB identifier) of the grouping object.

The URI of the service is composed of four parts:

- host name
- port
- service name
- resource

For example: http://localhost:27000/Connector/DataService/api/Hosts

- host name = "localhost"

- port = 27000
- service name = “Connector/DataService”
- resource = api/Hosts

The service provides the following URI’s:

- **URI for getting hosts within the network**

URI	http://localhost:27000/Connector/DataService/api/Hosts	Returns hosts available in the network

JSON template	[
	{
	"Name": ""
	}
	]
		Host name

Sample JSON data	[
	{
	"Name": "CS03-VZS"
	},
	{
	"Name": "CS18-VT"
	}
	]	

- **URI for getting projects of a host**

	http://localhost:27000/Connector/DataService/api/Hosts/localhost	Returns projects of the given host (localhost)

JSON template	[
	{
	"DBName": "",
	"Name": "",
	"Code": ""
	}
	]
		Database name of the project
		Name of the project
		Code of the project

Sample JSON data	[{ "DBName": "438741d6-9a58-4ef3-8cca-afd0537b8958", "Name": "test1", "Code": "" }, { "DBName": "620e9cc3-548f-44d6-9d7b-cd88790891ab", "Name": "Stage 4", "Code": "stage4" },]	

- **URI for accessing CIM resources data**

URI	http://localhost:27000/Connector/DataService/api/DataView/Resource/Association/RId?vpsHost=Host&projectDBName=DbName&projectCode=Code	Returns data related to the given record id for the specified resource
Resource	The following resources can be specified: • *TagSystem* • *LBS* • *SchedulingSystem* • *CADModelSystem* • *TakeOffSystem* • *CostPlan* • *WorkPackageSystem* • *ConstructabilitySystem*	
Association	Specifies association of the returned data with the given record id (*RId*):	
	• *Item*	Returns the specified *RId* record and its properties
	• *Parents*	Returns the specified *RId* record with all parent records up to the root record (*RId* = *“0”*) and their properties
	• *Children*	Returns the specified *RId* record and its direct children records and their properties
	• *ChildrenRecursively*	Returns the specified *RId* record with all children records and their properties
Rid	Record Id is an auto-generated unsigned long number. 0 is always the root record id and all requests should start with this ID.	
Host	The VPS host name	
projectDBName projectCode	Either project database name or project code should be specified If *projectDBName* is used *projectCode* can be skipped altogether If *projectCode* is used *projectDBName* should be left empty	
JSON template	{	
	"header": {	Groups VPS and project specific data
	"vpsHost": "",	Host name
	"projectDbName": "",	Project database name
	"projectCode": ""	Project code
	}, "messages": { "infos": [], "warnings": [], "errors": [], "authenticationErrors": [], "authorizationErrors": [] },	Groups server messages by type generated during the request
	"views": [	Holds a single view of the requested resource

	{	
	"viewType": "",	Resource name
	"guid": "",	Unique ID of the view
	"records": [{	Ordered list of records; the requested *RId* record is always the first record in the list; all records share the same six attributes
	"rid": "325",	Record Id is an auto-generated unsigned long
	"prid": "297",	Record Id of the parent record
	"loid": "1003.0.421493",	Logical DB identifier of the object of the record; This identifier is unique within the VPS
	"type": "COMPONENT",	Type of the record; Check above for all record types
	"isDeleted": "0",	0 if the record is not deleted; 1 if it was deleted
	"isExpandable": "1",	0 if the record has no children; 1 if it has children and can be expanded
	"properties": [{	A list of record specific properties; all properties share the same four attributes;
	"pid": "0",	Property Id is an unsigned integer number
	"name": "code",	String name of the property
	"dataType": "String",	Server data type of property;
	"value": "A1010"	Serialized string value of the property

	},
	{
	"group": {
		Grouping, if applicable, groups one or more properties that share a common associated object: e.g. location

	"type": "byLoc",	Logical name of the group
	"caption": "1003.0.118899",	A common attribute of the associated object – e.g. its loid.

	"properties": [
	{
	"pid": "101",
	"name": "srcQty",
	"dataType": "Double",
	"value": "1.0"
	}
	]
	},
	{
	"group": {
	"type": "byLoc",
	"caption": "1003.0.112233",
	"properties": [
		Properties of the group have the same attributes as the regular properties of the record

```
                                    {
                                        "pid": "102",
                                        "name": "srcQty",
                                        "dataType": "Double",
                                        "value": "1.0"
                                    }
                                ]
                            }
                        }
                    ]
                }
            ]
        }
    ]
}
```

Sample JSON data

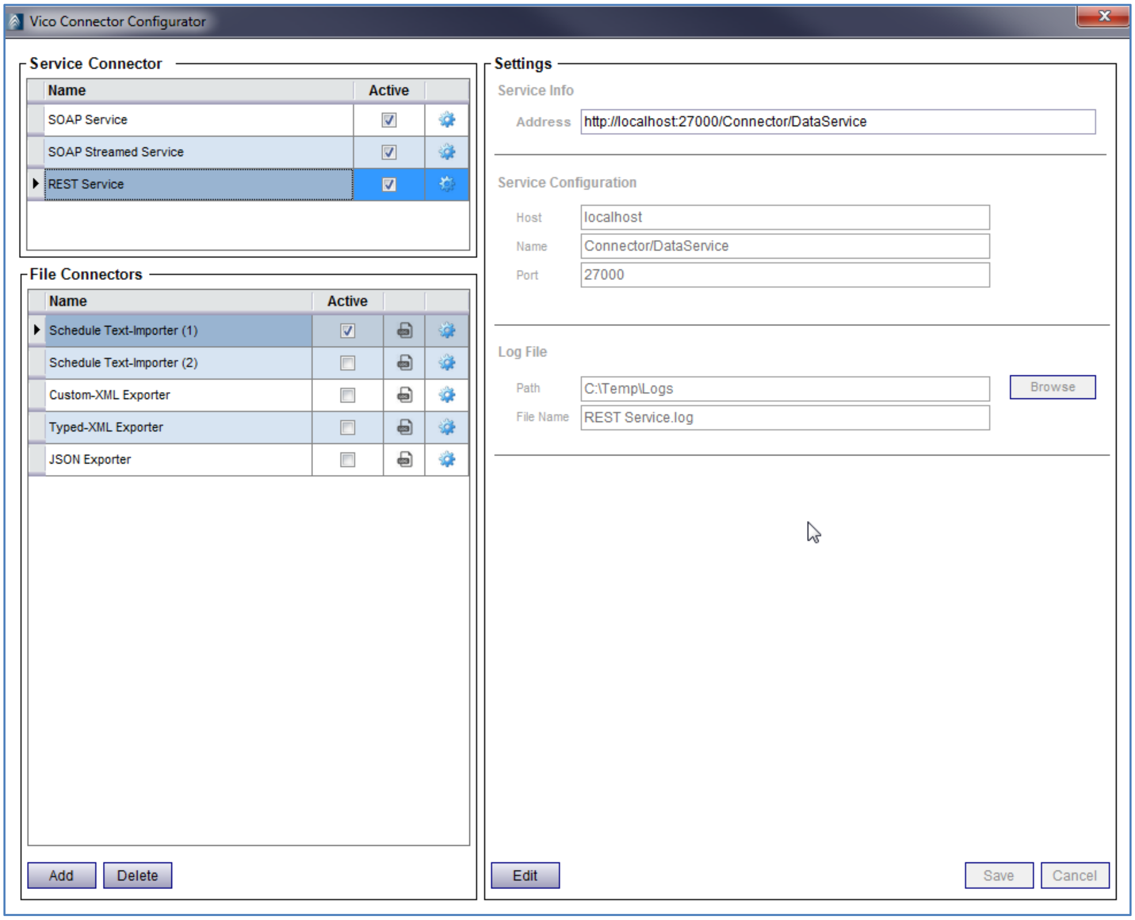
```
{
    "header": {
        "vpsHost": "localhost",
        "projectDbName": "620e9cc3-548f-44d6-9d7b-cd88790891ab",
        "projectCode": "stage4"
    },
    "messages": {
        "infos": [],
        "warnings": [],
        "errors": [],
        "authenticationErrors": [],
        "authorizationErrors": []
    },
    "views": [
        {
            "viewType": "TagSystem",
            "guid": "C2DCDDB2-A76C-48CC-B881-462180663273",
            "records": [
                {
                    "rid": "10",
                    "prid": "3",
                    "loid": "1003.0.4169",
                    "type": "TAG",
                    "isDeleted": "0",
                    "isExpandable": "1",
                    "properties": [
                        {
                            "pid": "0",
                            "name": "name",
                            "dataType": "String",
                            "value": "CostType"
                        },
                        {
                            "pid": "1",
                            "name": "desc",
                            "dataType": "String",
                            "value": "CostType values"
                        }
                    ]
                },
                {
                    "rid": "11",
                    "prid": "10",
                    "loid": "1003.0.4169",
                    "type": "TAGVALUEHEADER",
                    "isDeleted": "0",
                    "isExpandable": "1"
                },
                {
                    "rid": "26",
                    "prid": "11",
                    "loid": "1003.0.4846",
                    "ploid": "0.0.0",
                    "type": "TAGVALUE",
                    "isDeleted": "0",
                    "isExpandable": "1",
                    "properties": [
                        {
                            "pid": "0",
                            "name": "name",
                            "dataType": "String",
                            "value": "Material"
                        },
                        {
                            "pid": "1",
                            "name": "desc",
                            "dataType": "String",
                            "value": ""
                        },
                        {
                            "pid": "2",
                            "name": "userDataMarkup",
                            "dataType": "Double",
                            "value": "10.0"
                        }
                    ]
                }
            ]
        }
```

	] }	

Vico Connector Service & Configurator

Vico Connector Service is a windows service being responsible for hosting the web services and the scheduled file import/export service. It is started automatically on system startup and is configured through Vico Connector Configurator, a small UI tool loaded in the system tray. Vico Connector Configurator can be brought up from the system-tray by right-clicking on its icon and selecting *Show* action.

The user interface has to parts: on the left the available service and file connectors are listed; on the right the selected connector's settings are presented. Each connector line shows the name, activation status and a gear icon ⚙ that opens the connector's general log dialog. This dialog presents connector-specific logs ordered by date, latest being on top. In addition, file connectors have a file icon 🗎 that opens import or export specific file log dialogs.



Service Connector

Name	Active	
SOAP Service	☑	⚙
SOAP Streamed Service	☑	⚙
REST Service	☑	⚙

File Connectors

Name	Active		
Schedule Text-Importer (1)	☑	🗎	⚙
Schedule Text-Importer (2)	☐	🗎	⚙
Custom-XML Exporter	☐	🗎	⚙
Typed-XML Exporter	☐	🗎	⚙
JSON Exporter	☐	🗎	⚙

Settings

Service Info

Address: http://localhost:27000/Connector/DataService

Service Configuration

Host: localhost
Name: Connector/DataService
Port: 27000

Log File

Path: C:\Temp\Logs
File Name: REST Service.log

Service Connectors

Service connectors offer basic configuration options for the available web services: buffered SOAP service, streamed SOAP service and a REST Service. They are automatically activated when VOWS is installed. Basic settings like connector activation, service host, name, port and log file can be changed by editing the connector. Any change will take place after saving the changes. The service's address can be found on the top of the settings pane.

File Connectors

File connectors offer a means to import or export data from or to files on a scheduled basis. Five predefined file connectors are installed with VOWS, each showing different usage scenarios. The sample import connector ships with importable files and a sample project, check within: \Program Files\Vico Software\VOWS\Vico Office Services. Unlike service connectors, which are fixed, file connectors can be added or deleted.

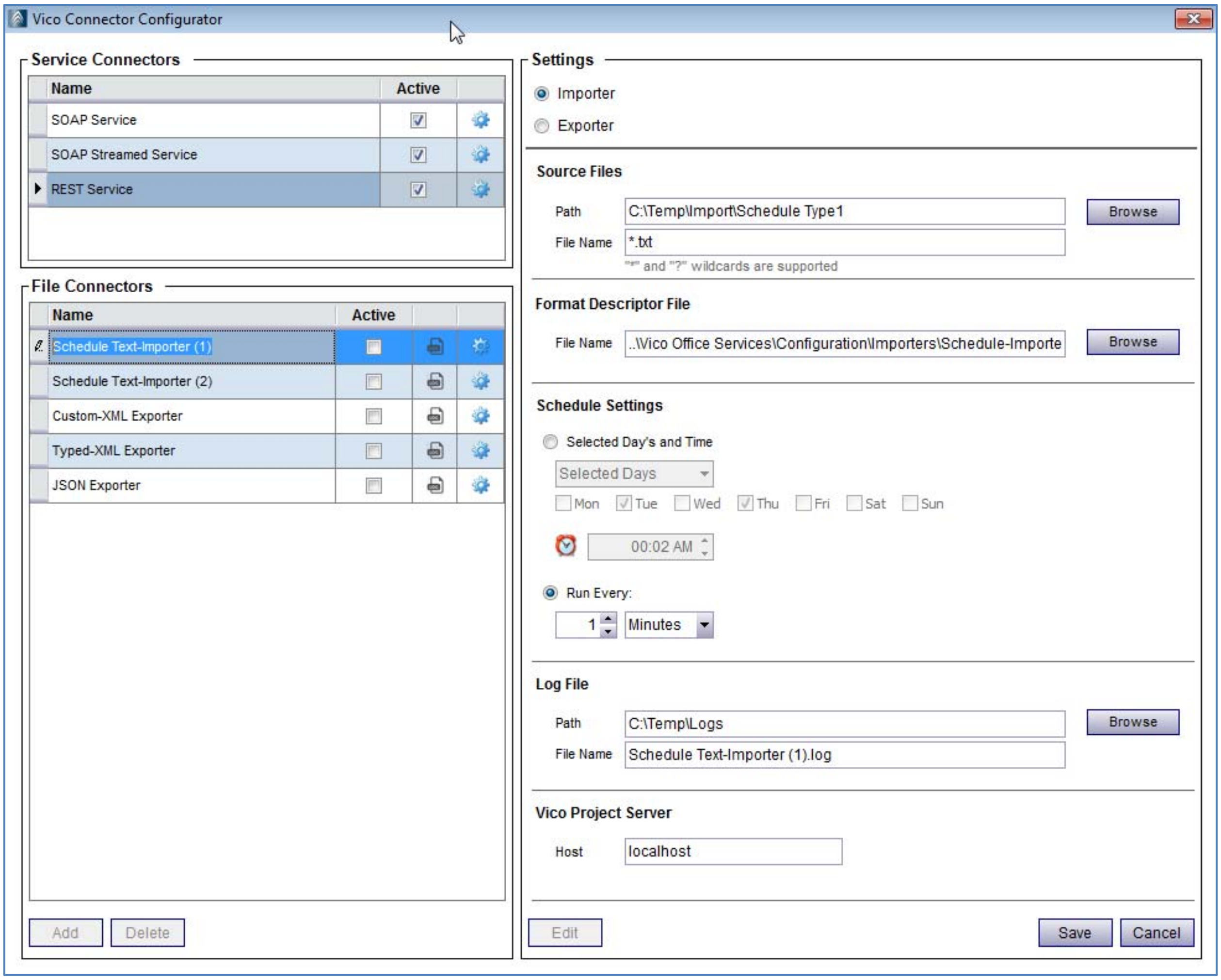
Import Connector

Import connector settings are divided in five groups, two of which are import specific:

Source Files: Files that should be imported are searched here

- **Path** (required): Path of the files to be imported
- **File Name** (required): Can be a fixed name or a name with wildcards to match multiple files. Two wildcards can be set:
    - Asterisk (*): Use the asterisk as a substitute for zero or more characters in a name
    - Question mark (?): Use the question mark as a substitute for a single character in a name

Vico Project Server (optional): specifies a valid VPS host from the LAN where the files will be imported. If the imported file already contains VPS host information this field should be left empty.



Vico Connector Configurator

Service Connectors

Name	Active
SOAP Service	☑
SOAP Streamed Service	☑
REST Service	☑

File Connectors

Name	Active
Schedule Text-Importer (1)	□
Schedule Text-Importer (2)	□
Custom-XML Exporter	□
Typed-XML Exporter	□
JSON Exporter	□

Settings

● Importer
○ Exporter

Source Files

Path: C:\Temp\Import\Schedule Type1
File Name: *.txt
"*" and "?" wildcards are supported

Format Descriptor File

File Name: ..\Vico Office Services\Configuration\Importers\Schedule-Importe

Schedule Settings

○ Selected Day's and Time
Selected Days
□ Mon ☑ Tue □ Wed ☑ Thu □ Fri □ Sat □ Sun
00:02 AM

● Run Every: 1 Minutes

Log File

Path: C:\Temp\Logs
File Name: Schedule Text-Importer (1).log

Vico Project Server

Host: localhost

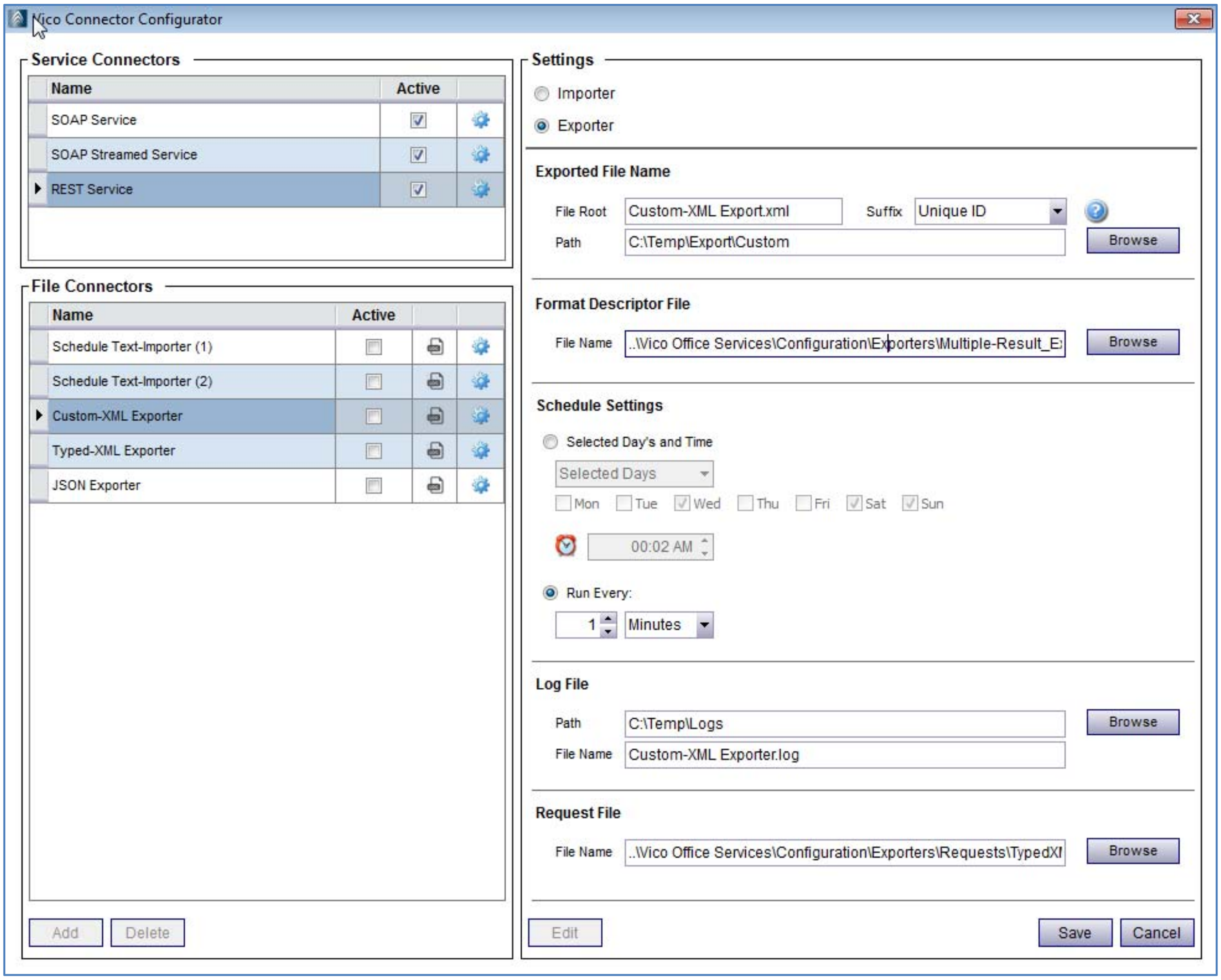
Export Connector

Export connector settings are divided in five groups, two of which are export specific:

Exported File Name: Specifies a path and a template for the name of the exported file

- **Path** (required): Path of the exported files
- **File Root** (required): Template name of the file. The name will be appended with the selected suffix: *Date Stamp* or *Unique ID*

Request File (required): Contains the request information. For more information, see Data Exchange XML Format – Request Document



Vico Connector Configurator

Service Connectors

Name	Active	
SOAP Service	☑	⚙
SOAP Streamed Service	☑	⚙
▶ REST Service	☑	⚙

File Connectors

Name	Active		
Schedule Text-Importer (1)	□	🖨	⚙
Schedule Text-Importer (2)	□	🖨	⚙
▶ Custom-XML Exporter	□	🖨	⚙
Typed-XML Exporter	□	🖨	⚙
JSON Exporter	□	🖨	⚙

Settings

○ Importer
● Exporter

Exported File Name

File Root: Custom-XML Export.xml Suffix: Unique ID
Path: C:\Temp\Export\Custom

Format Descriptor File

File Name: ..\Vico Office Services\Configuration\Exporters\Multiple-Result_E:

Schedule Settings

○ Selected Day's and Time
Selected Days
□ Mon □ Tue ☑ Wed □ Thu □ Fri ☑ Sat ☑ Sun
00:02 AM

● Run Every:
1 Minutes

Log File

Path: C:\Temp\Logs
File Name: Custom-XML Exporter.log

Request File

File Name: ..\Vico Office Services\Configuration\Exporters\Requests\TypedXI

Common Connector Settings

Schedule Settings: Lets you define a schedule to run the connector.

- Selected Day(s) and Time: *Daily*, *Day of Month* and *Selected Days* can be chosen for the day/days of run. The timer sets the time of the day the connector is run
- Run Every: Sets the time interval between 1 to 60 Minutes or 1 to 24 Hours that the connector is run

Log File (optional): Specifies the path and name for a file where connector specific-information is logged. Logged information is shown in either the connector's general log dialog ⚙ or the import log dialog 🖨.

Format Descriptor File (optional): Contains the description of a custom format, as described in File Format Transformations below, which is used to transform custom text or XML files into Data Exchange XML format.

- File Name (optional): Relative or absolute path to a format descriptor file.

  If the file name is left empty: the imported file must have a valid Data Exchange XML format and the exported file will have a Data Exchange XML or JSON format

File Format Transformations

An important feature of a file connector is its capability to import files with custom text or XML format and to export to a custom XML format. Exporting to custom text format is not supported. The format descriptor file is an XML file containing format-specific definitions.

Custom XML to Data Exchange XML Transformation

XML to XML transformation can be used by both import and export connectors. The configuration file contains a single *XMLFormat* element (see Data Structures). The actual format transformation is done by the specified XSL transformation. It is out of the scope of this document to describe how XSLT works; therefore, a good understanding of both XSLT and Data Exchange XML structure is mandatory.

Sample XML to XML configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>

<XMLFormat xsi:noNamespaceSchemaLocation="..\..\Schemas\Connector\v1.0\Service\FormatTypes.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <xsltPath>.\Configuration\Importers\XSLT\custom_tags.xsl</xsltPath>

  <xsltParams>

    <param>

      <name> costTypeTagValues </name>

      <value> 0;SocialExpenses|1;Labor|2;Material|3;Subcontractor|4;Equipment|5;Other </value>

    </param>

  </xsltParams>

</XMLFormat>
```

Custom Text to Data Exchange XML Transformation

Text to XML transformation can be used by import connectors only. The configuration file contains a single *TextFormat* element (see Data Structures). Text format transformation is done in two steps: first, a text to raw XML transformation, then a raw XML to Data Exchange XML transformation by the specified XSL transformation.

The information in the text file must be in a de-normalized form: each line should contain all the information needed to import the data in that line. Two types of text formats are supported:

- Column based: This format has data in distinct columns delimited by a specified character (e.g. tab or comma)
- Indexed: This format has known and fixed length data at specified indexes

If *delimiter* element is not empty, column-based format is assumed.

Sample line of column based format (tab used as delimiter):

```
Ctrl01  01-SUB-003     2810.0.64281   2957.0.20528   20140707       3       100
```

Sample line of indexed based format (*in the text file, this is one line*):

```
0000    1 000Ctrl0120140801PA    00007072014EUR010000000kkp 2014/8      01-SUB-0030000000 00007
2810.0.64281                 00000000828595  2957.0.20528                  T      00000000000003
0052020     00010000
```

Step 1 – Text to raw XML conversion is done based on the specified format mappings. The *mapping**s*** element specifies a 1:1 correlation between a single text line and a single raw XML element. A single *mapping* element is mapped to an XML node in the raw XML and defines the following:

Sample mapping element and the XML node generated out of it:

`<mapping required="Y">`
`        <sourceIndex>1</sourceIndex>`
`        <dataType>String</dataType>`
`        <targetNode>projectCode v="{0}"/</targetNode>`
`</mapping>`
	`<projectCode v="{0}"/>`

- Is required: If the *required* attribute equals "*Y*", it specifies that this data is mandatory for importing the line
    - Warning message: If data is missing, a warning message of *"[Node*] is missing"* gets logged. *Node** is the name of the mapping XML node e.g. "*projectCode*"
- Data: The actual information replacing "*{0}*" placeholder
    - Can be a constant value, e.g. "*/schTask*"
    - A column-based format specifies the *sourceIndex* of the column that contains the information; *sourceIndex* starts from 1.
    - An indexed-based format specifies the *startPosition* and *charLength* of the information within a line.
- Data Type: Validates that the de-serialized data has the specified type. Four types are available:
    - *Decimal*: used together with *TextFormat*'s *decimalSymbol* and *digitGroupingSymbol* to convert data to a decimal type
    - *DateTime:* used together with *TextFormat*'s *dateTimeFormat* to convert data to a date-time type
    - *Boolean*: convert data to a Boolean type. Possible values are "0", "1", "false" or "true"
    - *String*: data is by default a string so no conversion is done
    - Warning message: if conversion fails a warning message of *"[Node*] has invalid 'Type*' value"* is added to the log. *Node** is the name of the mapping XML node e.g. "*date*"; *Type** the data type
- Target Node: Forms the template of an actual XML node. The placeholder string "*{0}*" is replaced by data and the whole string is prepended by '<' and appended by '>' characters to form a real XML node; e.g. "*date v="{0}"/*" becomes "<*date* v='20140707'/>"

Step 2 – Raw XML to Data Exchange XML conversion is basically an XML to XML conversion. For information about this conversion see, Custom XML to Data Exchange XML Transformation.

Example 1 – Configuring a Column Based Format Descriptor File

The problem: It is given a text file that contains information about latest progress entries of location tasks in lines with tab character separated columns.

Steps to be taken in order to define the format descriptor file:

A. Each line should contain all the information needed by the import. First, confront the corresponding Data Exchange XML data structure with information within the line: shown below is an XML excerpt of a task structure with latest progress entry. Beside project code, which is in the header element, all the information needed for importing is shown.

```
<schTask loid="2810.0.64281" ploid="0.0.0" key="01-SUB-003">
        <locTasks>
                <locTask loid="2957.0.20528">
                        <latestProgressEntry>
                                <date v="2014-07-07"/>
                                <type v="FINISH"/>
                                <completion v="1"/>
                        </latestProgressEntry>
                </locTask>
        </locTasks>
</schTask>
```

Assure that the given text line contains the required information:

```
Ctrl01  01-SUB-003     2810.0.64281   2957.0.20528   20140707       3       100
```

- Project code: `Ctrl01`
- Schedule task loid or key attribute: both are present: `2810.0.64281` and `01-SUB-003`
- Location task loid: `2957.0.20528`
- Latest progress entry information:
    - Date in a specific format: `20140707` as yyyyMMdd
    - Type: as defined in Data Structures is an enumeration of *ProgressTypeEnum* type; however, in the text line this is specified as a number, “3” in our example. A mapping between *ProgressTypeEnum* items and numbers should be specified. Mappings can be configured in step 2, XML to XML transformation. Therefore, a mapping between numbers and *ProgressTypeEnum* values will be passed to the XSL transformation engine by specifying a *param* in *xsltParams*:

```
<xsltParams>
        <param>
```

```
                <name>typeValuePairs</name>

                <value>0;BEGIN|1;CONTINUE|2;PAUSE|3;FINISH</value>

        </param>

</xsltParams>
```

When defining the XSL transformation, use this parameter's value to map the number to an accepted *latestProgressEntry/type* value.

- Completion: "100" is specified as percentage while XML uses fractions so XSL transformation shall divide raw XML's completion with 100

If the line contains all the required information, proceed to next step. If required information is missing from the line, then it cannot be imported; it should be checked if the format can be extended with the missing information.

B. Define the raw XML and the mapping items. By default the text parser inserts the current line number as an attribute of the first XML node.

```
<schTask line="14">

        <loid v="2810.0.64281"/>

        <projectCode v="Ctrl01"/>

        <key v="01-SUB-003"/>

        <locTask>

                <loid v="2957.0.20528"/>

                <latestProgressEntry>

                        <date v="2014-07-07"/>

                        <type v="3"/>

                        <completion v="100"/>

                </latestProgressEntry>

        </locTask>

</schTask>
```

#1	#2	#3	#4	#5	#6	#7
Ctrl01	01-SUB-003	2810.0.64281	2957.0.20528	20140707	3	100

`<schTask line="14">`	`<mapping>`
	`        <constValue>schTask</constValue>`
	`        <dataType>String</dataType>`

	<targetNode>{0}</targetNode> </mapping>
<loid v="2810.0.64281"/>	<mapping required="Y"> <sourceIndex>3</sourceIndex> <dataType>String</dataType> <targetNode>loid v="{0}"/</targetNode> </mapping>
<projectCode v="Ctrl01"/>	<mapping required="Y"> <sourceIndex>1</sourceIndex> <dataType>String</dataType> <targetNode>projectCode v="{0}"/</targetNode> </mapping>
<key v="01-SUB-003"/>	<mapping> <sourceIndex>2</sourceIndex> <dataType>String</dataType> <targetNode>key v="{0}"/</targetNode> </mapping>
<locTask>	<mapping> <constValue>locTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
<loid v="2957.0.20528"/>	<mapping required="Y"> <sourceIndex>4</sourceIndex> <dataType>String</dataType> <targetNode>loid v="{0}"/</targetNode> </mapping>
<latestProgressEntry>	<mapping> <constValue>latestProgressEntry</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>

<date v="2014-07-07"/>	<mapping> <sourceIndex>5</sourceIndex> <dataType>DateTime</dataType> <targetNode>date v="{0}"/</targetNode> </mapping>
<type v="3"/>	<mapping> <sourceIndex>6</sourceIndex> <dataType>String</dataType> <targetNode>type v="{0}"/</targetNode> </mapping>
<completion v="100"/>	<mapping> <sourceIndex>7</sourceIndex> <dataType>Decimal</dataType> <targetNode>completion v="{0}"/</targetNode> </mapping>
</latestProgressEntry>	<mapping> <constValue>/latestProgressEntry</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
</locTask>	<mapping> <constValue>/locTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
</schTask>	<mapping> <constValue>/schTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>

C. Define file format and data type specific values. Note that *delimiter* element specifies the column separator character. If *delimiter* is not specified, indexed based format is assumed.

```
<dateTimeFormat>yyyyMMdd</dateTimeFormat>

<decimalSymbol>,</decimalSymbol>

<digitGroupingSymbol>.</digitGroupingSymbol>

<delimiter>\t</delimiter>
```

Example 2 – Configuring an Indexed-Based Format Descriptor File

The problem: it is given a text file that contains information about latest progress entries of location tasks in indexed based lines.

Steps to be taken in order to define the format descriptor file:

A. Each line should contain all the information needed by the import – first thing is to confront the corresponding Data Exchange XML data structure with information within the line: shown below is an XML excerpt of a task structure with latest progress entry. Beside project code, which is in the header element, all the information needed for importing is shown.

```
<schTask loid="2810.0.64281" ploid="0.0.0" key="01-SUB-003">

        <locTasks>

                <locTask loid="2957.0.20528">

                        <latestProgressEntry>

                                <date v="2014-07-07"/>

                                <type v="FINISH"/>

                                <completion v="1"/>

                        </latestProgressEntry>

                </locTask>

        </locTasks>

</schTask>
```

Assure that the given text line contains the required information:

```
0000    1 000Ctrl0120140801PA    00007072014EUR010000000kkp 2014/8      01-SUB-0030000000 00007
2810.0.64281                 00000000828595  2957.0.20528                 T       0000000000003
0052020     00010000
```

- Project code: `Ctrl01`
- Schedule task loid or key attribute: both are present: `2810.0.64281` and `01-SUB-003`
- Location task loid: `2957.0.20528`
- Latest progress entry information:
  - Date in a specific format: `07072014` as MMddyyy
  - Type: specifies progress type as a numbers, “3” in our example – see Example 1

- Completion: "100" is specified as percentage while XML uses fractions so XSL transformation shall divide raw XML's completion with 100

If the line contains all the required information, proceed to the next step. If required information is missing from the line, then it cannot be imported. It should be checked if the format can be extended with the missing information.

B. Define the raw XML and the mapping items. By default, the text parser inserts the current line number as an attribute of the first XML node.

```
<schTask line="14">

        <loid v="2810.0.64281"/>

        <projectCode v="Ctrl01"/>

        <key v="01-SUB-003"/>

        <locTask>

                <loid v="2957.0.20528"/>

                <latestProgressEntry>

                        <date v="2014-07-07"/>

                        <type v="3"/>

                        <completion v="100"/>

                </latestProgressEntry>

        </locTask>

</schTask>
```

Importable line	
`0000      1 000Ctrl0120140801PA     00007072014EUR010000000kkp 2014/8        01-SUB-0030000000 00007 2810.0.64281              00000000828595 2957.0.20528              T      00000000000003 0052020     00010000`	

`<schTask line="14">`	`<mapping>`
	`        <constValue>schTask</constValue>`
	`        <dataType>String</dataType>`
	`        <targetNode>{0}</targetNode>`
	`</mapping>`

`        <loid v="2810.0.64281"/>`	`<mapping required="Y">`
	`        <startPosition>100</startPosition>`

	<charLength>25</charLength> <dataType>String</dataType> <targetNode>loid v="{0}"/</targetNode> </mapping>
<projectCode v="Ctrl01"/>	<mapping required="Y"> <startPosition>17</startPosition> <charLength>6</charLength> <dataType>String</dataType> <targetNode>projectCode v="{0}"/</targetNode> </mapping>
<key v="01-SUB-003"/>	<mapping> <startPosition>76</startPosition> <charLength>10</charLength> <dataType>String</dataType> <targetNode>key v="{0}"/</targetNode> </mapping>
<locTask>	<mapping> <constValue>locTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
<loid v="2957.0.20528"/>	<mapping required="Y"> <startPosition>141</startPosition> <charLength>25</charLength> <dataType>String</dataType> <targetNode>loid v="{0}"/</targetNode> </mapping>
<latestProgressEntry>	<mapping> <constValue>latestProgressEntry</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>

<date v="2014-07-07"/>	<mapping> <startPosition>40</startPosition> <charLength>8</charLength> <dataType>DateTime</dataType> <targetNode>date v="{0}"/</targetNode> </mapping>
<type v="3"/>	<mapping> <startPosition>185</startPosition> <charLength>2</charLength> <dataType>String</dataType> <targetNode>type v="{0}"/</targetNode> </mapping>
<completion v="100"/>	<mapping> <startPosition>209</startPosition> <charLength>5</charLength> <dataType>Decimal</dataType> <targetNode>completion v="{0}"/</targetNode> </mapping>
</latestProgressEntry>	<mapping> <constValue>/latestProgressEntry</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
</locTask>	<mapping> <constValue>/locTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode> </mapping>
</schTask>	<mapping> <constValue>/schTask</constValue> <dataType>String</dataType> <targetNode>{0}</targetNode>

	`</mapping>`

C. Define the file format and data type specific values. Note that the *delimiter* element is missing. Therefore, an indexed based format is assumed.

```
<dateTimeFormat>MMddyyyy</dateTimeFormat>

<decimalPlaces>2</decimalPlaces>
```

Data Structures

Common Types

All XML Schema Definition files are within the folder \Program Files\Vico Software\VOWS\Vico Office Services*Schemas**DataExchange**v1.0*.

XML elements common to request, response and transmit documents can be found in: *Common**Types.xsd*.

XML elements common to resource data structures can be found in: \Program Files\Vico Software\VOWS\Vico Office Services*Schemas**DataExchange**v1.0**Cost_Management**Common**Types.xsd*.

Resource specific common XML types	
XML Type	Description
LoidType	A maximum 32 characters length *xs:string*
AssociationActionTypeEnum *values*:	Used during importing only. Specifies what action should be taken on the associated collection before the given items (if any) are processed: *Set* – Specified items are added to the current collection *ClearAndSet* – Collection is cleared first then specified items are added. If no items are specified, the collection is simply cleared.
*xxx*Field and xxxFieldRO Attributes: *v*, *ro*	Generic field definition where *xxx* can be replaced by a specific type: Boolean, Int, UInt, Decimal, Loid, Date • ***v****:* required attribute that holds the actual value of the field. - The *type* property sets the type of the field: e.g. *xs:boolean, xs:integer, xs:string, xs:unsignedInt, EmptyDateType, EmptyDateTimeType* - *use="required"* • ***ro****:* optional attribute that logically sets whether the field is read-only or not. It is used in the XSD schema to distinguish between read-only and writable fields. At import the given TransmitDocument based XML is logically validated to contain only fields that have *ro="N"* attributes by default. • *xxx*Field has *ro="N"* by default

	• xxxFieldRO has *ro=”Y”* by default
xxxOrEmptyField Attributes: *v, ro*	Similar to the above beside that: • ***v:*** is an optional attribute that has *use=”optional”*
StringField and StringFieldRO Attributes: *v, ro, isKey*	Generic string field definition. *v* and *ro* attribute are similar to *xxx*Field • *isKey:* optional attribute that specifies whether this field is a key; thus, its value is unique within its collection.
StringFieldKey Attributes: *v, ro, isKey*	Generic string field definition. *v* and *ro* attribute are similar to *xxx*Field • *isKey=”Y”*
EmptyDateType EmptyDateTimeType	Simlpe type definition based on *xs:date* and *xs:dateTime*
LIRefType Attributes: *loid, key*	Base type for an item of a list structure • *loid:* required attribute that specifies the logical database identifier (loid) of the element • *key:* optional attribute, *use=”optional”*, it has the same value as the element’s field value marked with *isKey=’Y’*
TIRefType Attributes: *ploid, loid, key*	Base type for an item of a tree structure. *loid* and *key* attributes are similar to that of LIRefType • *ploid:* required attribute, holds the parent element’s *loid* attribute. The parent element must be defined in the XML before any other element references it.
TIRefByKeyType Attributes: *ploid, loid, key*	Similar to the above beside that: • *key:* required attribute, *use=”required”*
DynamicPropType Attributes: *loid*	Base type for a dynamic property item. Dynamic properties are created by means of an association of the record’s object with another object: e.g. Component *by* location properties (as quantity, price, total price). These ‘*by something*’ properties are grouped together within a *Group*. • *loid:* required attribute, specifies the associated object’s loid
MarkupValueTypeField Attributes: *v, ro*	Defines the value type of a markup • *v:* required attribute of type MarkupValueTypeEnum • *ro:* fixed=”N”
MarkupValueTypeEnum *values:*	The markup value can represent one of these values: *TagPercentage* – The value is from TagValType : userDataMarkup, see MarkupType note. *OverwritePercentage* – Manually given percentage value *OverwriteValue* – Manually given value

TTVRefListType

Attributes: *action* of type AssociationActionTypeEnum; Use optional, default=*”Set”*

Field Name	XML Type	Can Create | Can Update	Description
ttv	TTVPairType	No | Yes	Referenced tag – tagValue pairs

TTVPairType

Field Name	XML Type	R-O	Description
tagLoid	LoidType	No	Loid of a Tag
tagValLoid	LoidType	No	Loid of a Tag Value from the above Tag

MarkupListType

Attributes: *action* of type AssociationActionTypeEnum; Use optional, default=*"Set"*

Field Name	XML Type	Can Create | Can Update	Description
markup	MarkupType	Yes | Yes	Holds a sequence of MarkupType elements

MarkupType extends LIRefType – Note: This type is related to a TagValType, the *loid* attribute is a reference to a TagValType element

Field Name	XML Type	R-O	Description
valueType	MarkupValueTypeField	No	Value type of the markup
value	DecimalField	No	Overwriten value of a markup. This field can represent a percentage or a value depending on the *valueType* field

ElementType extends LIRefType

Field Name	XML Type	R-O	Description
elemID	StringFieldRO	Yes	Internal CAD element ID
revitElemID	StringFieldRO	Yes	Revit element ID if the element is from a Revit model or empty

ElementListType

Field Name	XML Type	Can Create | Can Update	Description
elem	ElementType	No | No	Markup

Cost Plan

The XML Schema Definition for this data structure is: \Cost_Management\Cost_Plan\\Cost_Plan.xsd.

CostPlanType

Field Name	XML Type	Description
addonList	AddonListType	Holds a sequence of AddonType elements
componentTreeList	ComponentsListType	Holds a sequence of ComponentType elements

AddonListType

Field Name	XML Type	Can create | Can Update	Description
addon	AddonType	Yes | Yes	Holds a sequence of AddonType elements

AddonType extends LIRefType

Field Name	XML Type	R-O	Description
code	StringField	No	Code of the addon

desc	StringField	No	Description of the addon
markupType	AddonMarkupTypeField	No	An addon can be a percentage of the addon calculation value (Net Total) or a fixed value added to the net total. This field specifies whether *markupVal* or *markupPct* fields should be used to calculate the addon value
markupVal	DecimalField	No	On export it is the calculated value of the addon On import it is fix value for the addon
markupPct	DecimalField	No	The actual addon value will be this fractional number multiplied by net total
useNetTotal	BooleanField	No	Net Total (*ComponentType – netTot* field) is used in this addon's calculation
isActive	BooleanField	No	Is this addon active
isDiv	BooleanField	No	Is this addon divided
addonList	ReferencedAddonListType	-	Container for referenced addons used in calculating this addon's value

AddonMarkupTypeField Attributes: *v, ro*	This field specifies whether *markupVal* or *markupPct* fields should be used to calculate the addon value • *v:* required attribute of type AddonMarkupTypeEnum • *ro:* fixed="N"
AddonMarkupTypeEnu *values*:	*Percent – markupPct* should be used to calculate the addon value *Value – markupVal* should be used to calculate the addon value

ComponentsListType

Field Name	XML Type	Can Create | Can Update	Description
comp	ComponentType	Yes | Yes	Holds a sequence of ComponentType elements

ComponentType extends TIRefType

Field Name	XML Type	R-O	Description
code	StringFieldKey	No	Component code
desc	StringField	No	Component description
cnsmp	DecimalField	No	Consupmtion
cppa	DecimalFieldRO	Yes	Cost/Parent Assembly
netTot	DecimalFieldRO	Yes	Net Total
grTot	DecimalFieldRO	Yes	Gross Total
isActive	BooleanField	No	Is the component activated
level	UintFieldRO	Yes	Tree level of the component. Project level is 0
addon	DecimalField	No	Add-On value
isManualAddon	BooleanField	No	Specifies whether the component has manually set addon value. On importing a false (0) value the addon of the component will be

			reset
baseMarkupValue	DecimalFieldRO	Yes	Total base markup value
baseMarkupPct	DecimalFieldRO	Yes	Total base markup percentage
netMarkupValue	DecimalFieldRO	Yes	Total net markup value
netMarkupPct	DecimalFieldRO	Yes	Total net markup percentage
maxGrTot	DecimalFieldRO	Yes	Max gross total value
maxMarkupVal	DecimalFieldRO	Yes	Max markup value
maxBaseCost	DecimalFieldRO	Yes	Max base cost
maxUnitCost	DecimalField	No	Max unit cost
maxVarBaseTot	DecimalFieldRO	Yes	Max net total variance
maxVarUnitCost	DecimalFieldRO	Yes	Max unit cost variance
minGrTot	DecimalFieldRO	Yes	Min gross total value
minMarkupVal	DecimalFieldRO	Yes	Min markup value
minBaseCost	DecimalFieldRO	Yes	Min base cost
minUnitCost	DecimalField	No	Min unit cost
minVarBaseCost	DecimalFieldRO	Yes	Min base cost variance
minVarUnitCost	DecimalFieldRO	Yes	Min unit cost variance
baseCost	DecimalField	No	Base cost
paMaxBaseCost	DecimalFieldRO	Yes	Pre- assembly max base cost
paMaxUnitCost	DecimalField	No	Pre-assembly unit cost
paMinBaseCost	DecimalFieldRO	Yes	Pre-assembly min base cost
paMinUnitCost	DecimalField	No	Pre-assembly min unit cost
paBaseCost	DecimalFieldRO	Yes	Pre-assembly base cost
paUnitCost	DecimalField	No	Pre-assembly unit cost
pppa	DecimalFieldRO	Yes	%/Parent Assembly
qty	DecimalFieldRO	Yes	Quantity
targetType	ActiveTargetTypeField	No	Specifies the target type
targetCost	DecimalField	No	Target cost will be set if *targetType="Cost"*
targetRate	DecimalField	No	Target rate will be set if *targetType="Rate"*
unit	StringField	No	Unit/UOM
unitCost	DecimalField	No	Unit cost
uom	StringField	No	UOM
varBaseCost	DecimalFieldRO	Yes	Base cost variance

varUnitCost	DecimalFieldRO	Yes	Unit cost variance
waste	DecimalField	No	Waste
taskLoid	LoidField	No	Loid reference to a task object
taskSameAsParent	BooleanField	No	Is task same as parent
taskHpu	DecimalField	No	Task hours per unit
byLoc	CostValueByLocationType	-	Component related values groupped by a location
locations	ReferencedLocationsListType	-	Container for Component-Formula locations
addonList	ReferencedAddonListType	-	Container for addons distributed specificaly to this component
formula	FormulaType	-	Formula container
tags	TTVRefListType	-	Tag container
baseMarkupList	MarkupListType	-	Base markup list container
netMarkupList	MarkupListType	-	Net markup list container
CostValueByLocationType extends DynamicPropType			
Field Name	XML Type	R-O	Description
srcQty	DecimalFieldRO	Yes	Source quantity by location
qty	DecimalFieldRO	Yes	Quantity by location
compPrice	DecimalFieldRO	Yes	Component price (base cost) by location
totPrice	DecimalFieldRO	Yes	Total price (active price) by location
ObjectRefType extends LIRefType			
Field Name	XML Type	R-O	Description
loid	LoidType	No	Loid of an object
ReferencedAddonListType Attributes: *action* of type AssociationActionTypeEnum; Use optional, default=*"Set"*			
Field Name	XML Type	Can Create | Can Update	Description
addon	ObjectRefType	No | Yes	Referenced addon
ReferencedLocationsListType Attributes: *action* of type AssociationActionTypeEnum; Use optional, default=*"Set"*			
Field Name	XML Type	Can Create | Can Update	Description
loc	ObjectRefType	No | Yes	Referenced location
FormulaType Attributes: *action* of type AssociationActionTypeEnum; Use optional, default=*"Set"*			

Field Name	XML Type	R-O	Description
string	StringField	No	Formula string
srcQty	DecimalFieldRO	Yes	Source quantity
toffs	TOIQRefListType	-	Container for TOI – TOQ pairs from the formula

TOIQRefListType			
Field Name	XML Type	Can Create | Can Update	Description
toff	TOIQPairType	No | Yes	Represents a single TOI – TOQ used in the formula

TOIQPairType			
Field Name	XML Type	R-O	Description
toiLoid	LoidType	No	Loid of a TOI
toqLoid	LoidType	No	Loid of a TOQ

TOQModeTypeFieldRO Attributes: *v*, *ro*	Defines a takeoff quantity mode field • *v:* required attribute of type TOQModeType • *ro:* fixed=”Y”
TOQModeType *values:*	*Manual* – manual TOQ *ModelBased*– model based TOQ

ReferencedElementsListType			
Field Name	XML Type	Can Create | Can Update	Description
elem	ObjectRefType	No | No	Referenced element

ActiveTargetTypeField Attributes: *v*, *ro*	Defines the target cost type • *v:* required attribute of type ActiveTargetTypeEnum • *ro:* fixed=”Y”
TOQModeType *values:*	*Manual* – manually entered TOQ – Note that imported TOQs will be *Manual* by default *ModelBased*– model based TOQ

Takeoff System

The XML Schema Definition for this data structure is: \Cost_Management\\ *TakeOff_System\TakeOff_System.xsd*.

TakeOffSystemType		
Field Name	XML Type	Description
takeoffs	TakeOffListType	Container for takeoff items

TakeOffListType			
Field Name	XML Type	Can Create | Can Update	Description
toi	TOIType	Yes | Yes	Holds a sequence of TOIType elements

TOIType extends LIRefType

Attributes: *logicalType* of type ElemTypeEnum; Use required

Field Name	XML Type	R-O	Description
name	StringField	No	Name of the takeoff item
type	ElemTypeField	No	Element type of the takeoff item
toqs	TOQListType	-	Container for take off quantity elements of this TOI
elements	ElementsLocationsListType	-	Container that holds the elements of a TOI and the location these elements are distributed on
locations	LocationsElementsListType	-	Container that holds the locations TOI is distributed on and the TOI elements that belong to this location

TOQListType

Field Name	XML Type	Can Create | Can Update	Description
toq	TOQType	Yes | Yes	Holds a sequence of TOIType elements

TOQType extends LIRefType

Field Name	XML Type	R-O	Description
name	StringField	Yes	Code of the takeoff quantity
unit	StringFieldRO	Yes	Description of the takeoff quantity
value	DecimalFieldRO	No	Container for takeoff quantity elements of this TOI
type	TOQModeTypeFieldRO	Yes	Take off quantity mode: manual or model based
byLoc	TOQByLocationType	-	Container that holds the locations TOI is distributed on and the TOI elements that belong to this location

ElemTypeField Attributes: *v*, *ro*	Defines the takeoff item’s element type • *v:* required attribute of type ElemTypeEnum • *ro:* fixed=”N”
ElemTypeEnum *values:*	MANUAL, BEAM_RECTANGULAR, COLUMN_RECTANGULAR, CURTAIN_WALL, DOOR, LAMP, DUCT_RECTANGULAR, EQUIPMENTACCESSORIES, OBJECT, ROOF, ROOM, SLAB, STAIR, SURFACE, WALL, WINDOW, DUCT_ROUNDOVAL, PIPECONDUIT, BEAM_PROFILED, COLUMN_PROFILED, CURTAIN_WALL_FRAME, CURTAIN_WALL_PANEL, DUCT_FITTING, PIPE_FITTING, CABLE_TRAY, CABLE_TRAY_FITTING, RAILING

ElementsLocationsListType

Field Name	XML Type	Can Create | Can Update	Description
elem	ElementLocationsType	No | No	Holds a sequence of TOIType elements

ElementLocationsType extends ElementType

Field Name	XML Type	R-O	Description
locations	ReferencedLocationsListType	-	Container for location references – this element is split on these locations

ReferencedElementsListType

Field Name	XML Type	Can Create | Can Update	Description
elem	ObjectRefType	No | No	Referenced element

LocationsElementsListType

Field Name	XML Type	Can Create | Can Update	Description
loc	LocationElementsType	No | No	Holds a sequence of TOIType elements

LocationElementsType extends LIRefType

Field Name	XML Type	R-O	Description
elements	ReferencedElementsListType	-	Container for element references – the referenced elements are split on this location

TOQByLocationType extends DynamicPropType

Field Name	XML Type	R-O	Description
value	DecimalField	No	TOQ value by location

LBS System

The XML Schema Definition for this data structure is: \Cost_Management\LBS\LBS_System.xsd.

LBSType

Field Name	XML Type	Description
locationSystems	LocationSystemListType	Container for location system items
locations	LocationListType	Container for location items

LocationSystemListType

Field Name	XML Type	Can Create | Can Update	Description
locSys	LocationSystemType	Yes | Yes	Holds a sequence of location system type elements

LocationSystemType extends LIRefType

Field Name	XML Type	R-O	Description
name	StringField	No	Name of the location system

LocationListType

Field Name	XML Type	Can Create | Can Update	Description
loc	LocationType	Yes | Yes	Holds a sequence of location (WBS Node) elements

LocationType extends TIRefType

Field Name	XML Type	R-O	Description

name	StringFieldKey	No	Name of the location
Level	IntFieldRO	Yes	Location level. The root location level is 0
nodeType	WBSNodeTypeFieldRO	Yes	Location (WBS Node) type
WBSNodeTypeFieldRO Attributes: *v*, *ro*	Defines a read-only field for WBS Node type • *v:* required attribute of type WBSNodeTypeEnum • *ro:* fixed="Y"		
WBSNodeTypeEnum *values*:	NT_UNKNOWN, NT_ROOT, NT_STOREY, NT_ZONE, NT_LOGICAL, NT_TYPE		

Tag System

The XML Schema Definition for this data structure is: \Cost_Management\Tag_System\Tag_System.xsd.

TagSystemType			
Field Name	XML Type	Description	
tagCats	TagCategoryListType	Container for tag category items	
TagCategoryListType			
Field Name	XML Type	Can Create | Can Update	Description
tagCat	TagCategoryType	Yes | Yes	Holds a sequence of tag catergory type elements
TagCategoryType extends LIRefType			
Field Name	XML Type	R-O	Description
name	StringField	No	Name of the tag category
desc	StringField	No	Description of the tag category
tags	TagListType	-	Container for tag elements
TagListType			
Field Name	XML Type	Can Create | Can Update	Description
tag	TagType	Yes | Yes	Holds a sequence of tag elements
TagType extends LIRefType Attributes: *logicalType* of type TagTypeEnum; Use required			
Field Name	XML Type	R-O	Description
name	StringField	No	Name of the tag
desc	StringField	No	Description of the tag
tagvals	TagValListType	-	Container for tag value elements
TagTypeEnum *values:*	COST_TYPE, STATUS, USER, TRADES_INVOLVED, CATEGORY, APPROVAL_STATE, COLOR, TYPE, CP_MARKUP,		

	WP_MARKUP

TagValListType			
Field Name	XML Type	Can Create | Can Update	Description
tv	TagValType	Yes | Yes	Holds a sequence of tag value elements

TagValType extends LIRefType			
Field Name	XML Type	R-O	Description
name	StringField	No	Name of the tag value
desc	StringField	No	Description of the tag value
userDataMarkup	DecimalField	No	The MarkupType uses this field

CAD Model System

The XML Schema Definition for this data structure is: *\Cost_Management\CAD_Model_System\CAD_Model_System.xsd*.

ConstructabilitySystemType		
Field Name	XML Type	Description
cadModels	CADModelListType	Container for cad model items

CADModelListType			
Field Name	XML Type	Can create | Can Update	Description
cadModel	CadModelType	No | No	Holds a sequence of CadModelType elements

CadModelType extends LIRefType			
Field Name	XML Type	R-O	Description
modelID	StringFieldRO	Yes	Identifier of the Model
isActive	BooleanFieldRO	Yes	Is this CAD model active
hasEBActive	BooleanFieldRO	Yes	Has this CAD model ever been activated
isDefault	BooleanFieldRO	Yes	Is the default CAD model
source	ModelSourceFieldRO	Yes	Source of the CAD model
activeVersionLoid	LoidFieldRO	Yes	The active model version's loid
versions	ModelVersionListType	-	Container that holds the model versions of this CAD model

ModelSourceFieldRO Attributes: *v*, *ro*	Defines a source model type field • *v:* required attribute of type ModelSourceEnum • *ro:* fixed="N"
ModelSourceEnum *values:*	Unknown, Revit, Tekla Structures, ArchiCAD, AutoCAD, CAD-Duct, IFC, SketchUp, DWG, Bentley, AutoCAD

	Note: AutoCAD is in the enumeration twice wrongly

ModelVersionListType

Field Name	XML Type	Can Create | Can Update	Description
modelVersion	ModelVersionType	No | No	Holds a sequence of model version type elements

ModelVersionType extends LIRefType

Field Name	XML Type	R-O	Description
desc	StringFieldRO	Yes	Model version description
isActive	BooleanFieldRO	Yes	Is this model version active
hasEBActive	BooleanFieldRO	Yes	Has this model version ever been activated
version	IntFieldRO	Yes	Version number
elems	OrigElementListType	-	Container that holds the original elements of this model version

OrigElementListType

Field Name	XML Type	Can Create | Can Update	Description
elem	OrigElementType	No | No	Holds a sequence of original element types

OrigElementType extends ElementType

Field Name	XML Type	R-O	Description
parts	ElementListType	-	Container that holds the derived elements that belong to the original element

Constructability System

The XML Schema Definition for this data structure is: *\Cost_Management\Constructability_System\Constructability_System.xsd*.

ConstructabilitySystemType

Field Name	XML Type	Description
issues	CMIssueListType	Container for constructabiliy issue type items
rfis	CMRFIListType	Container for request for information type items

CMIssueListType

Field Name	XML Type	Can create | Can Update	Description
issue	IssueItemType	Yes | Yes	Holds a sequence of constructability issue elements

IssueItemType extends CMItemType

Note: all fields are inherited from CMItemType

CMRFIListType

Field Name	XML Type	Can create | Can Update	Description
rfi	RFIItemType	Yes | Yes	Holds a sequence of request for information elements
RFIItemType extends CMItemType			
Note: all fields are inherited from CMItemType			
CMItemType extends LIRefType			

Field Name	XML Type	R-O	Description
code	StringFieldKey	No	Constructability management item (CM item) code
location	StringField	No	Location name
dateCreated	DateTimeFieldOrEmptyField	No	Creation date
priority	PriorityTypeField	No	Item priority
desc	StringField	No	Description of the item
owner	StringField	No	Owner
status	StatusTypeField	No	Item status
elementType	StringFieldRO	Yes	The types of the elements related to this item
elementIDs	StringFieldRO	Yes	The ID's of the elements related to this item
referencedModel	StringFieldRO	Yes	The referenced models of the elements related to this item
referencedDoc	StringFieldRO	Yes	The documents referenced by this item
linkedDoc	StringFieldRO	Yes	The documents linked to this item
type	IssueTypeField	No	Item type
costImpact	StringField	No	Cost impact of the change
timeImpact	StringField	No	Time impact of the change
dateRequired	DateTimeFieldOrEmptyField	No	Required date
requestedBy	StringField	No	Name of the person(s) who requested the changes
gridReference	StringField	No	Grid reference
assumption	StringField	No	Assumption about change
subject	StringField	No	Subject of the item
request	StringField	No	Request
suggestion	StringField	No	Suggestion
response	StringField	No	Response
tags	TTVRefListType	-	Tag container
discussion	DiscussionType	-	Discussion elements container
viewPoints	ViewPointListType	-	View point elements container
images	ImageListType	-	Image element container

DiscussionType			
Field Name	XML Type	Can Create | Can Update	Description
message	InstantMessageType	Yes | Yes	Container that holds instant messages of an item

InstantMessageType extends LIRefType			
Field Name	XML Type	R-O	Description
dateCreated	DateTimeFieldOrEmptyField	Yes	Creation date of the instant message
dateModified	DateTimeFieldOrEmptyField	Yes	Last modified date
owner	StringField	Yes	Owner
text	StringField	Yes	Text of the instant message
isVisible	BooleanField	Yes	Is visible

ViewPointListType			
Field Name	XML Type	Can Create | Can Update	Description
viewPoint	ViewPointType	No | No	Container that holds the view points of an item

ViewPointType extends LIRefType			
Field Name	XML Type	R-O	Description
name	StringFieldRO	Yes	View point name
isDefault	BooleanFieldRO	Yes	Is the default view point
screenShotImageData	StringFieldRO	Yes	Image data in Base64 format

ImageListType			
Field Name	XML Type	Can Create | Can Update	Description
image	ImageType	Yes | Yes	Container that holds the images of an item

ImageType extends LIRefType			
Field Name	XML Type	R-O	Description
data	StringField	No	Image data in Base64 format
isDefault	BooleanField	No	Is the default image
zoomRatio	DecimalField	No	Image's zoom ratio

PriorityTypeField Attributes: *v*, *ro*	Defines a cost management issue priority type field • *v:* required attribute of type PriorityTypeEnum • *ro:* fixed="N"
PriorityTypeEnum *values*:	Low, Medium, High, Top, None
StatusTypeField Attributes: *v*, *ro*	Defines a cost management issue status type field • *v:* required attribute of type StatusTypeEnum

	- *ro:* fixed="N"
StatusTypeEnum *values*:	New, Pending, Reviewed, InProgress, Resolved, Unknown
IssueTypeField Attributes: *v*, *ro*	Defines a cost management issue type field - *v:* required attribute of type IssueTypeEnum - *ro:* fixed="N"
IssueTypeEnum *values*:	Clash, Manual, Cloud, Undefined

Scheduling System

The XML Schema Definition for this data structure is: *\Cost_Management\Scheduling_System\Scheduling_System.xsd*.

SchedulingSystemType

Field Name	XML Type	Description
tasks	TaskListType	Container for task items

TaskListType

Field Name	XML Type	Can create | Can Update	Description
schTask	ScheduleTaskType	Yes | Yes	Holds a sequence of schedule task elements
sumTask	SumTaskType	Yes | Yes	Holds a sequence of summary task elements

ScheduleTaskType extends CIMTaskBaseType
Attributes: *ploid* of type LoidType; Use required

Field Name	XML Type	R-O	Description
locSysLoid	LoidField	No	Location sytem Loid
actualProductivity	DecimalFieldRO	Yes	Actual productivity
subTasks	SubTaskListType	-	Container for subtasks (Task Part – Schedule Planner terminology)
detTasks	DetailTaskListType	-	Container for detail location tasks

SumTaskType extends CIMTaskBaseType
Attributes: *ploid* of type LoidType; Use required

Field Name	XML Type	R-O	Description
locSysLoid	LoidField	No	Location sytem loid

CIMTaskBaseType extends LIRefType

Field Name	XML Type	R-O	Description
code	StringFieldKey	No	Task code

name	StringField	No	Task name
plannedStartDate	DateTimeFieldRO	Yes	Planned start date and time
plannedEndDate	DateTimeFieldRO	Yes	Planned end date and time
forecastedStartDate	DateTimeFieldRO	Yes	Forecasted start date and time
forecastedEndDate	DateTimeFieldRO	Yes	Forecasted end date and time
SubTaskListType			
Field Name	XML Type	Can Create | Can Update	Description
subTask	SubTaskType	No | Yes	Container for subtasks
LocationTaskListType			
Field Name	XML Type	Can Create | Can Update	Description
locTask	LocationTaskType	No | Yes	Container for location tasks
SubTaskType extends CIMTaskBaseType			
Field Name	Field Name	R-O	Field Name
actualProductivity	DecimalFieldRO	Yes	Actual productivity
supplier	StringFieldRO	Yes	Supplier
locTasks	LocationTaskListType	-	Container for location tasks
crew	CrewType	Yes	Task crew
LocationTaskType extends CIMTaskBaseType			
Field Name	XML Type	R-O	Description
locLoid	LoidFieldRO	Yes	Location loid
hasBegun	BooleanFieldRO	Yes	Has begun
actualProductivity	DecimalFieldRO	Yes	Actual productivity
latestProgressEntry	ProgressEntryType	Yes	Latest progress entry (actual) state **[Used only for import]**
progressEntries	ProgressEntryListType	-	Container for all progress entries
CrewType			
Field Name	XML Type	R-O	Description
count	IntegerFieldRO	Yes	Crew count
members	MemberListType	-	Container for all crew members
DetailTaskListType			
Field Name	XML Type	Can Create | Can Update	Description
detTask	DetailTaskType	No | Yes	Container for detail location tasks
DetailTaskType extends CIMTaskBaseType			
Field Name	XML Type	Can Create | Can Update	Description

actualProductivity	DecimalFieldRO	No | Yes	Actual productivity
detSubTasks	DetailSubTaskListType	No | Yes	Container for detail subtasks
DetailSubTaskListType			
Field Name	XML Type	Can Create | Can Update	Description
detSubTask	DetailSubTaskType	No | Yes	Container for detail subtasks
DetailLocationTaskListType			
Field Name	XML Type	Can Create | Can Update	Description
detLocTask	DetailLocationTaskType	No | Yes	Container for detail location tasks
ProgressEntryListType			
Field Name	XML Type	Can Create | Can Update	Description
progressEntry	ProgressEntryTypeRO	No | No	Container for detail location tasks
MemberListType			
Field Name	XML Type	Can Create | Can Update	Description
member	MemberType	No | No	Container for detail location tasks
DetailSubTaskType extends CIMTaskBaseType			
Field Name	XML Type	R-O	Description
actualProductivity	DecimalFieldRO	Yes	Actual productivity
supplier	StringFieldRO	Yes	Supplier
detLocTasks	DetailLocationTaskListType	-	Container for detail location tasks
crew	CrewType	Yes	Task crew
DetailLocationTaskType extends CIMTaskBaseType			
Field Name	XML Type	R-O	Description
locLoid	LoidFieldRO	Yes	Location loid
hasBegun	BooleanField	Yes	Has the task begun
actualProductivity	DecimalFieldRO	Yes	Actual productivity
latestProgressEntry	ProgressEntryType	Yes	Latest progress entry (actual) state **[Used only for import]**
progressEntries	ProgressEntryListType	-	Container for all progress entries
MemberType extends LIRefType			
Field Name	XML Type	R-O	Description
code	StringFieldRO	Yes	Code
name	StringFieldRO	Yes	Name
prodFactor	DecimalFieldRO	Yes	Production factor

quantity	DecimalFieldRO	Yes	Quantity
unitCost	DecimalFieldRO	Yes	Unit cost
ProgressEntryTypeRO extends LIRefType			
Field Name	XML Type	R-O	Description
date	DateFieldRO	Yes	Date
type	ProgressTypeFieldRO	Yes	Type
completion	DecimalFieldRO	Yes	Completion
isLatest	BooleanFieldRO	Yes	Is latest progress entry
ProgressEntryType			
Field Name	XML Type	R-O	Description
date	DateField	No	Date
type	ProgressTypeField	No	Type
completion	DecimalField	No	Completion
ProgressTypeField Attributes: *v*, *ro*	Defines the actual progress type of a task • *v:* required attribute of type ProgressTypeEnum • *ro:* fixed=”N”		
ProgressTypeEnum *values*:	NONE, BEGIN, PAUSE, CONTINUE, PROGRESS, FINISH		

Work Package System

The XML Schema Definition for this data structure is: *\Cost_Management\Work_Package_System\Work_Package_System.xsd*.

WorkPackageSystemType			
Field Name	XML Type	Description	
wpTreeList	WPTreeListType	Container for work package items	
WPTreeListType			
Field Name	XML Type	Can create | Can Update	Description
wp	WorkPackageType	Yes | Yes	Holds a sequence of work package elements
WorkPackageType extends TIRefByKeyType			
Field Name	XML Type	R-O	Description
code	StringFieldKey	No	Code
desc	StringOrEmptyField	No	Description

isSummary	BooleanOrEmptyField	No	Is summary work package
monitUnit	StringOrEmptyField	No	Monitoring unit
markupType	WPMarkupTypeField	No	Markup type
markupVal	DecimalField	No	Markup value
markupPct	DecimalField	No	Markup percentage
actuals	ActualListType	-	Container for actuals

ActualListType			
Field Name	XML Type	Can Create | Can Update	Description
actual	ActualType	Yes | Yes	Holds a sequence of actual elements

ActualType extends LIRefType			
Field Name	XML Type	R-O	Description
system	StringOrEmptyField	No	System
docID	StringOrEmptyField	No	Document identifier
dueDate	DateField	No	Due date
receivedDate	DateField	No	Receive date
desc	StringOrEmptyField	No	Description
costTypeTagValue	StringOrEmptyField	No	System / Cost Type Tag-Value name
sum	DecimalField	No	Sum
linkedDoc	PathOrEmptyStringFieldType	No	Linked document

PathOrEmptyStringFieldType Attributes: *v*, *ro*	Defines the actual progress type of a task • *v:* attribute of type PathType • *ro:* fixed="N"
PathType	A maximum 1024 characters length *xs:string.*
WPMarkupTypeField Attributes: *v*, *ro*	Defines the actual progress type of a task • *v:* attribute of type WPMarkupTypeEnum • *ro:* fixed="N"
WPMarkupTypeEnum *values*:	OverwritePercentage, OverwriteValue

VPS Info

The XML Schema Definition for this data structure is: *\Cost_Management\VPS_Info\VPS_Info.xsd*.

VPSType

Field Name	XML Type	Description
vpsList	VPSHostListType	Container for VPS host items

VPSHostListType

Field Name	XML Type	Can create | Can Update	Description
vpsHost	VPSHostType	No | No	Holds a sequence of VPS host elements

VPSHostType

Attributes: *name* of type UrlType; *use="required"*

Field Name	XML Type	Can create | Can Update	Description
projects	ProjectsListType	No | No	Container for project items

ProjectsListType

Attributes: *name* of type UrlType; *use="required"*

Field Name	XML Type	Can create | Can Update	Description
proj	ProjectType	No | No	Holds a sequence of project elements

ProjectType

Attributes: *dbName* of type ProjectDbName; *use="required"* – the database name of a project is a GUID

Field Name	XML Type	R-O	Description
code	ProjectCode	Yes	Code of the project – Vico Office\Dashboard – Code column
name	ProjectName	Yes	Name of the project
type	xs:string	Yes	Type
created	xs:dateTime	Yes	Created date
lastEdited	xs:dateTime	Yes	Last edited date
isAvailable	xs:boolean	Yes	Is available

UrlType, ProjectDbName, ProjectCode, ProjectName	A maximum 255 characters length *xs:string.*

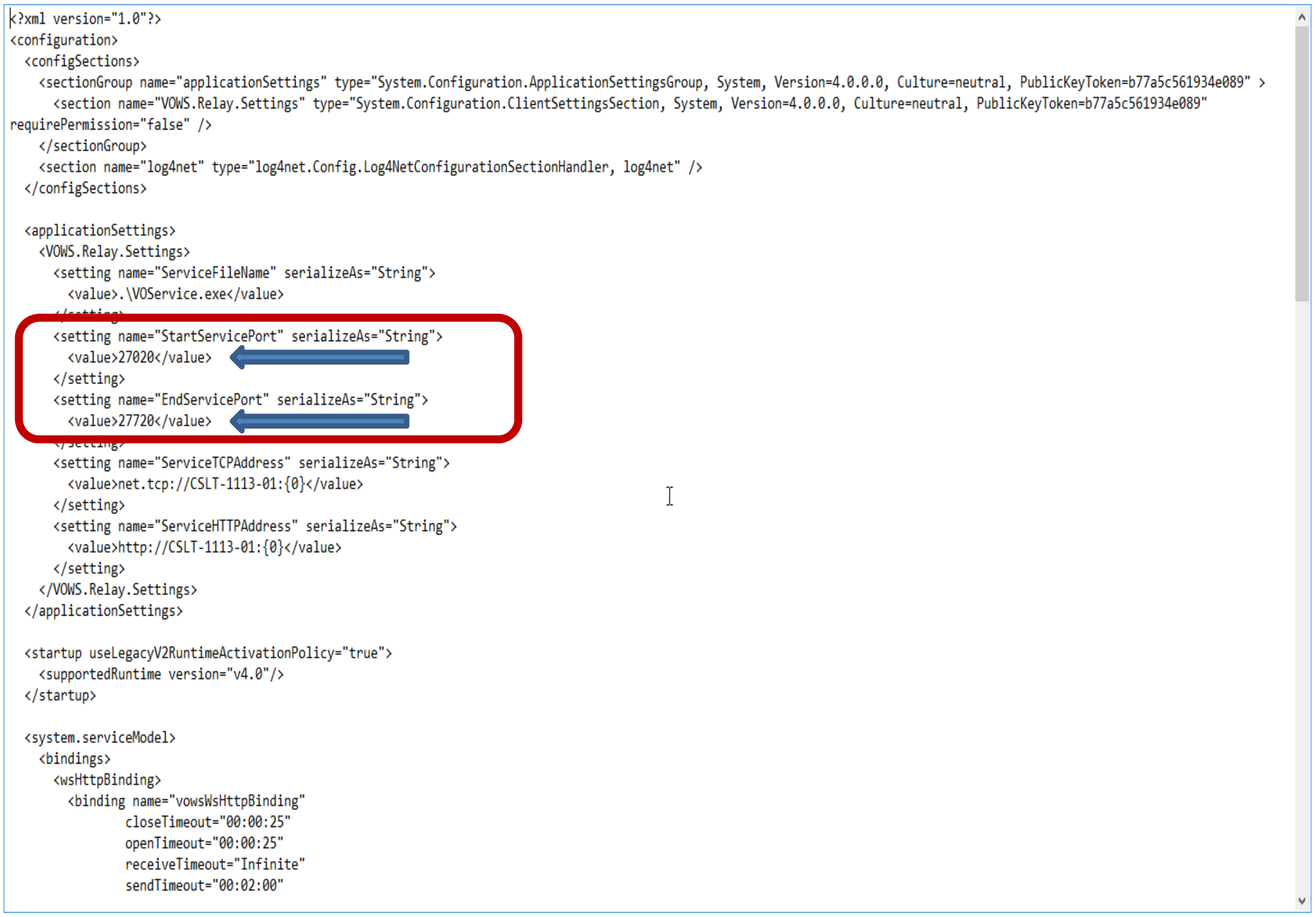
Using VOWS

Firewall

For the proper functioning of Vico Office Web Services a range of ports has to be opened in the firewall. These ports can be found in the C:\Program Files\Vico Software\VOWS\Vico Office Services\VOServiceRelay.exe.config file and in Vico Connector Configurator at each **Service Connector**'s Port field.

VOServiceRelay.exe.config

Open *VicoServiceRelay.exe.config* file to view or modify the range of ports as seen below:



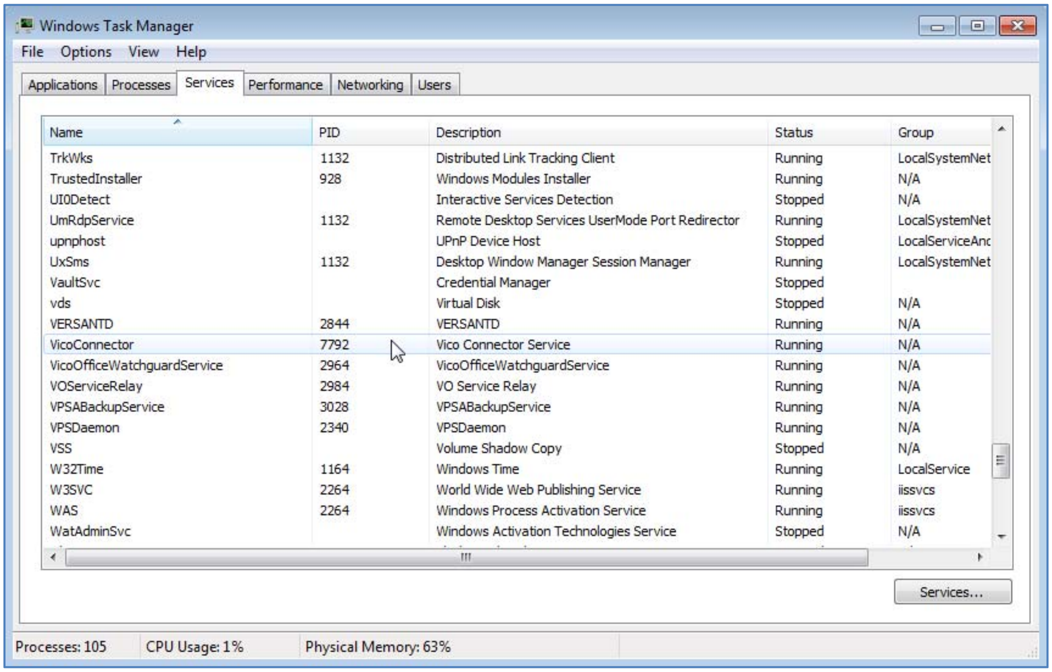
```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings" type="System.Configuration.ApplicationSettingsGroup, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section name="VOWS.Relay.Settings" type="System.Configuration.ClientSettingsSection, System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
    </sectionGroup>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net" />
  </configSections>

  <applicationSettings>
    <VOWS.Relay.Settings>
      <setting name="ServiceFileName" serializeAs="String">
        <value>.\VOService.exe</value>
      </setting>
      <setting name="StartServicePort" serializeAs="String">
        <value>27020</value>
      </setting>
      <setting name="EndServicePort" serializeAs="String">
        <value>27720</value>
      </setting>
      <setting name="ServiceTCPAddress" serializeAs="String">
        <value>net.tcp://CSLT-1113-01:{0}</value>
      </setting>
      <setting name="ServiceHTTPAddress" serializeAs="String">
        <value>http://CSLT-1113-01:{0}</value>
      </setting>
    </VOWS.Relay.Settings>
  </applicationSettings>

  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0"/>
  </startup>

  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="vowsWsHttpBinding"
                closeTimeout="00:00:25"
                openTimeout="00:00:25"
                receiveTimeout="Infinite"
                sendTimeout="00:02:00"
```

The default ports that VOWS will try to use are from 27020 through 27720. VOWS will use a port from this range only after assuring it is not already in use.

Troubleshooting VOWS

VOWS is based on two Windows Services: **VicoConnector** and **VOServiceRelay**. For a proper functioning both have to be running. In some cases the installer cannot start these services so these have to be started manually.

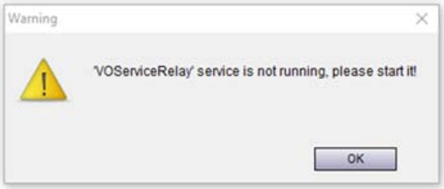
Check in Windows Task Manager that both services are running.



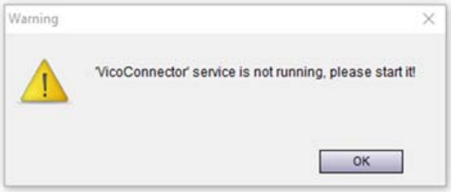
Name	PID	Description	Status	Group
TrkWks	1132	Distributed Link Tracking Client	Running	LocalSystemNet
TrustedInstaller	928	Windows Modules Installer	Running	N/A
UI0Detect		Interactive Services Detection	Stopped	N/A
UmRdpService	1132	Remote Desktop Services UserMode Port Redirector	Running	LocalSystemNet
upnphost		UPnP Device Host	Stopped	LocalServiceAnc
UxSms	1132	Desktop Window Manager Session Manager	Running	LocalSystemNet
VaultSvc		Credential Manager	Stopped	
vds		Virtual Disk	Stopped	N/A
VERSANTD	2844	VERSANTD	Running	N/A
VicoConnector	7792	Vico Connector Service	Running	N/A
VicoOfficeWatchguardService	2964	VicoOfficeWatchguardService	Running	N/A
VOServiceRelay	2984	VO Service Relay	Running	N/A
VPSABackupService	3028	VPSABackupService	Running	N/A
VPSDaemon	2340	VPSDaemon	Running	N/A
VSS		Volume Shadow Copy	Stopped	N/A
W32Time	1164	Windows Time	Running	LocalService
W3SVC	2264	World Wide Web Publishing Service	Running	iissvcs
WAS	2264	Windows Process Activation Service	Running	iissvcs
WatAdminSvc		Windows Activation Technologies Service	Stopped	N/A

Processes: 105 CPU Usage: 1% Physical Memory: 63%

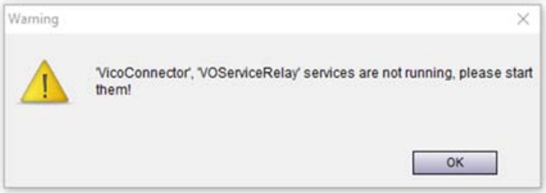
If at least one of the services is stopped a warning dialog is popped up when Vico Connector Configurator is being shown for the first time. In order to emphasize that something is not working properly Vico Connector Configurator's user interface will switch to a read-only mode (all buttons will be disabled). Once both services are started Vico Connector Configurator's user interface will enter into "normal" editable mode.



'VOServiceRelay' service is not running, please start it!



'VicoConnector' service is not running, please start it!



'VicoConnector', 'VOServiceRelay' services are not running, please start them!

Install VOWS services manually

If the Vico Office installer ran successfully but the VOWS services are not running (as indicated in the images above), the user will be able to install them manually.

1. Go the VOWS installation directory, which is by default in:

   ```
   C:\Program Files\Vico Software\VOWS
   ```

2. Open a command window in 'Vico Office Services' within the VOWS directory.

   Tip: To open a command window in Windows Explorer: Press SHIFT + right mouse click above the directory and select the 'Open command window here' option from the context menu. (Make sure the user how is doing this has *administrator privileges*.)

3. Locate the 'InstallUtil.exe'.

   Note: This tool was installed with the .NET Framework. If your Windows installation directory is C:\Windows, the tool is installed in this default path:

   ```
   C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe
   ```

   Since VOWS installation contains only 64-bit executables, we will need the path of the 64-bit version of .NET Framework 4 or 4.5.*.

4. Copy the InstallUtil.exe path in the command window and add the VOWS executable name as a parameter one by one.

The following commands need to be executed:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe VicoConnector.exe
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe VOServiceRelay.exe
```

Note: If the installation still fails with errors, please contact your administrator.

Uninstall VOWS services manually

1. Repeat the steps 1, 2 and 3 from the install section.

2. Copy the `InstallUtil.exe` path in the command window and add the VOWS executable name as a parameter one by one, but with and extra option.

The following commands needs to be executed:

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /u VicoConnector.exe
```

```
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe /u VOServiceRelay.exe
```

Tip: After the service executable is deleted, the service might still be present in the registry. To resolved that case too, open the command window and use the *sc* command to remove the entry for the service from the registry.

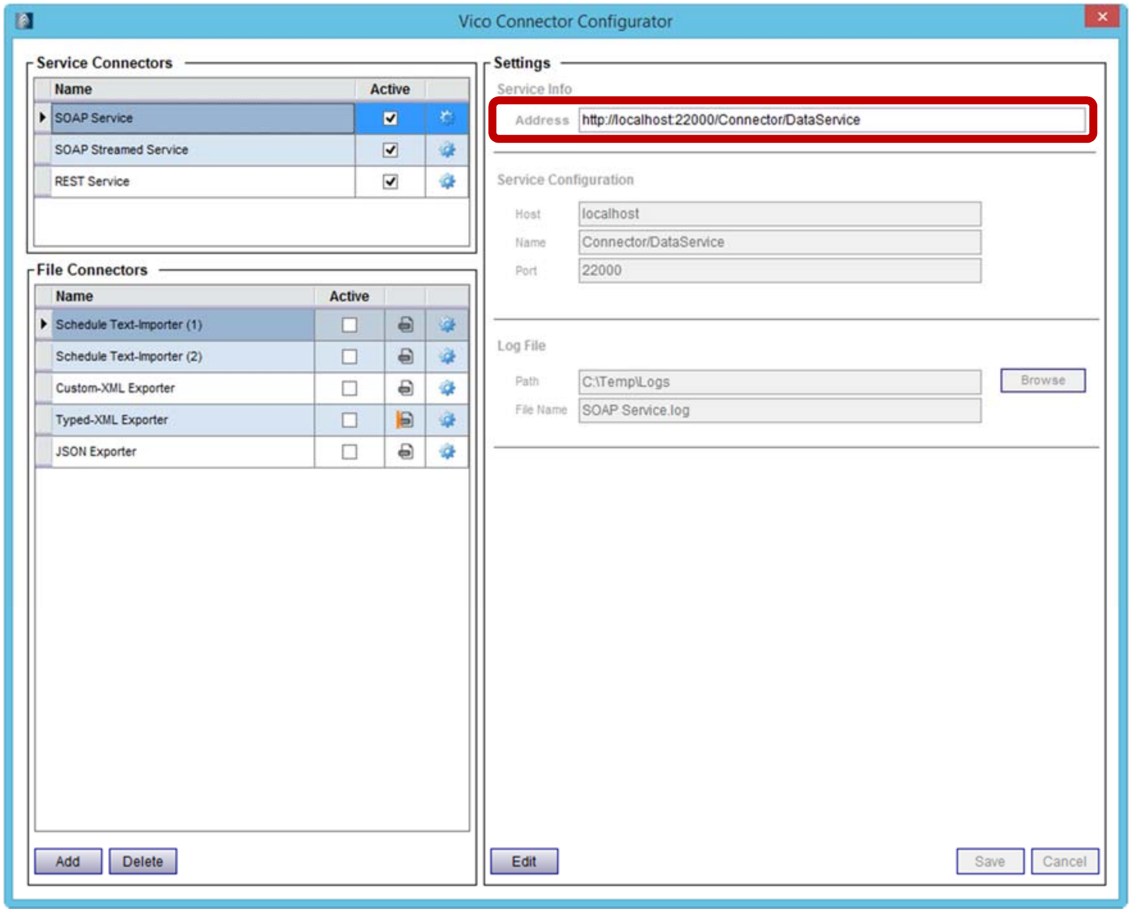
Example:

```
sc delete VicoConnector
sc delete VOServiceRelay
```

VicoConnector - Vico Connector Service

Vico Connector Service is a windows service being responsible for hosting the web services and the scheduled file import/export service.

In order to check that the XML web services are up-n-running copy the address highlighted below into a web browser:



Vico Connector Configurator

Service Connectors

Name	Active
SOAP Service	☑
SOAP Streamed Service	☑
REST Service	☑

File Connectors

Name	Active
Schedule Text-Importer (1)	☐
Schedule Text-Importer (2)	☐
Custom-XML Exporter	☐
Typed-XML Exporter	☐
JSON Exporter	☐

Settings

Service Info

Address: http://localhost:22000/Connector/DataService

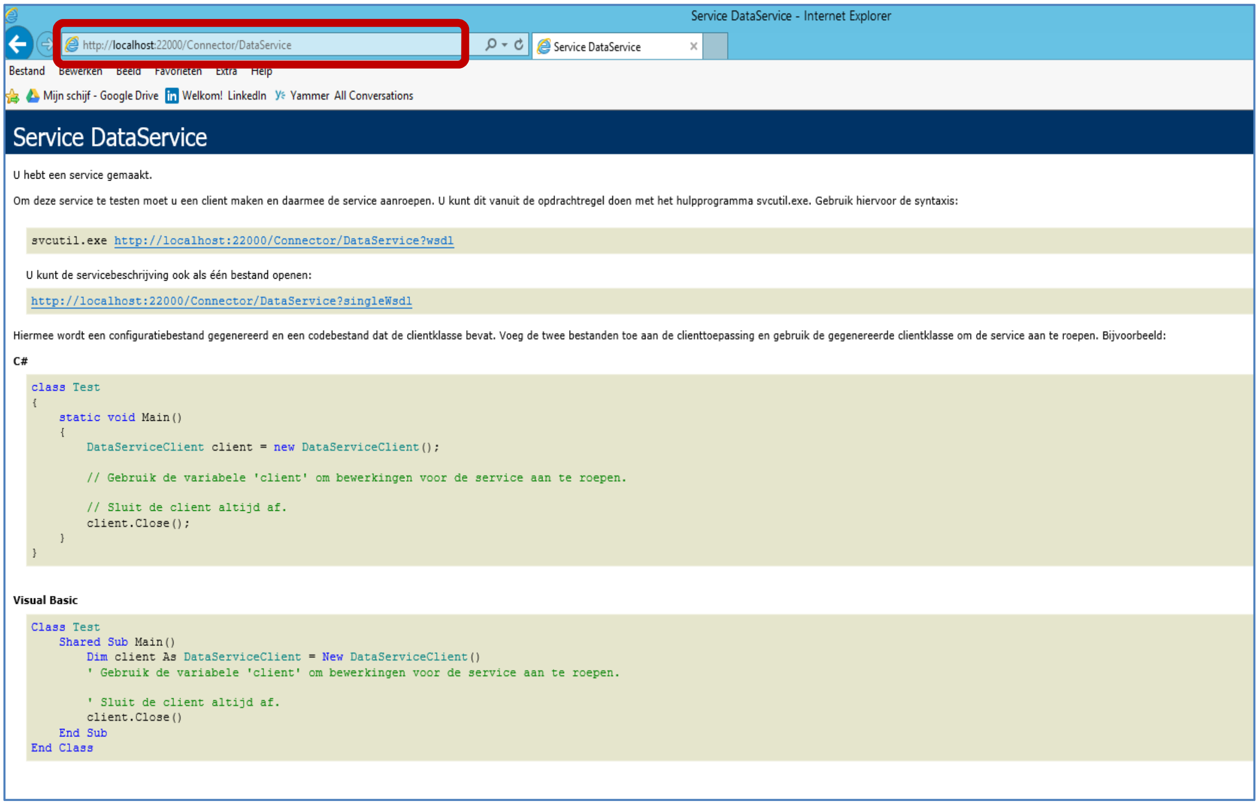
Service Configuration

Host: localhost
Name: Connector/DataService
Port: 22000

Log File

Path: C:\Temp\Logs
File Name: SOAP Service.log

If the SOAP Service **is running correctly, the following page should be displayed in the web browser:**


Service DataService - Internet Explorer
http://localhost:22000/Connector/DataService
Service DataService
Bestand Bewerken Beeld Favorieten Extra Help
Mijn schijf - Google Drive
Welkom! LinkedIn
Yammer All Conversations

Service DataService

U hebt een service gemaakt.

Om deze service te testen moet u een client maken en daarmee de service aanroepen. U kunt dit vanuit de opdrachtregel doen met het hulpprogramma svcutil.exe. Gebruik hiervoor de syntaxis:

```
svcutil.exe http://localhost:22000/Connector/DataService?wsdl
```

U kunt de servicebeschrijving ook als één bestand openen:

```
http://localhost:22000/Connector/DataService?singleWsdl
```

Hiermee wordt een configuratiebestand gegenereerd en een codebestand dat de clientklasse bevat. Voeg de twee bestanden toe aan de clienttoepassing en gebruik de gegenereerde clientklasse om de service aan te roepen. Bijvoorbeeld:

C#

```
class Test
{
    static void Main()
    {
        DataServiceClient client = new DataServiceClient();

        // Gebruik de variabele 'client' om bewerkingen voor de service aan te roepen.

        // Sluit de client altijd af.
        client.Close();
    }
}
```

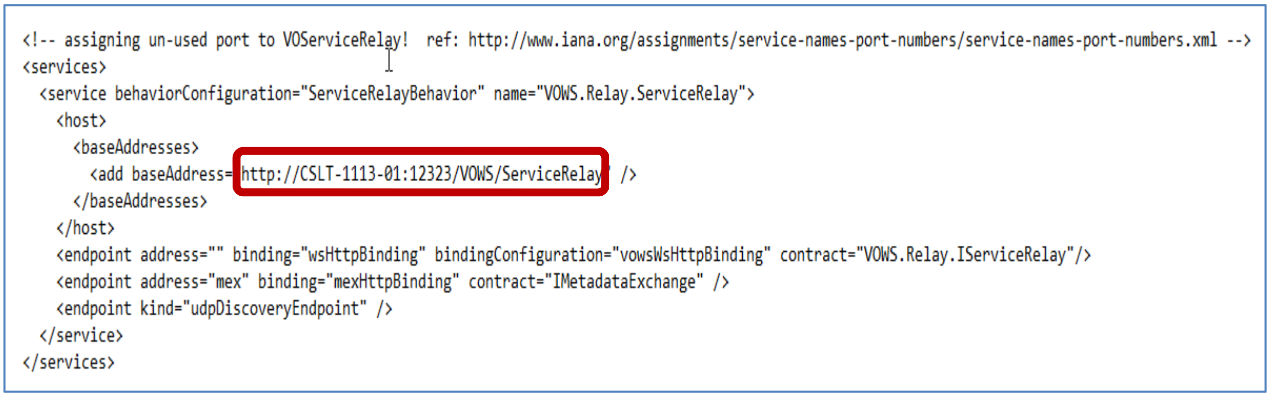
Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As DataServiceClient = New DataServiceClient()
        ' Gebruik de variabele 'client' om bewerkingen voor de service aan te roepen.

        ' Sluit de client altijd af.
        client.Close()
    End Sub
End Class
```

If this page does not appear the **VicoConnector** service should be restarted. This can be done through [Windows / Task Manager / Services](#).

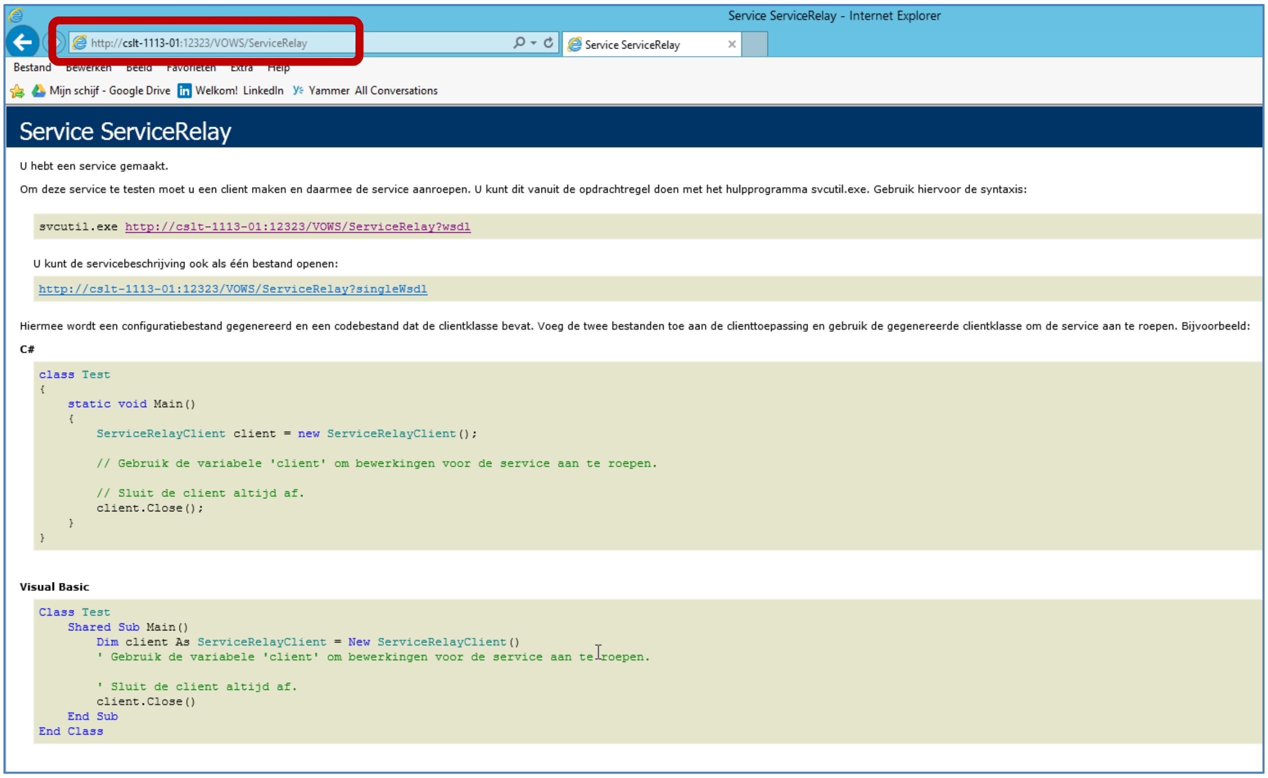
VOServiceRelay – VO Service Relay

In order to check that VOServiceRelay is working properly open *V*OServiceRelay.exe.config file and copy the highlighted string as shown below from baseAddress into a web browser. Please note that this address might differ on your machine.



```
<!-- assigning un-used port to VOServiceRelay!  ref: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml -->
<services>
  <service behaviorConfiguration="ServiceRelayBehavior" name="VOWS.Relay.ServiceRelay">
    <host>
      <baseAddresses>
        <add baseAddress="http://CSLT-1113-01:12323/VOWS/ServiceRelay" />
      </baseAddresses>
    </host>
    <endpoint address="" binding="wsHttpBinding" bindingConfiguration="vowsWsHttpBinding" contract="VOWS.Relay.IServiceRelay"/>
    <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
    <endpoint kind="udpDiscoveryEndpoint" />
  </service>
</services>
```

If the service is running properly, the following page should be displayed in the web browser:



Service ServiceRelay - Internet Explorer
http://cslt-1113-01:12323/VOWS/ServiceRelay
Service ServiceRelay
Bestand Bewerken Beeld Favorieten Extra Help
Mijn schijf - Google Drive
Welkom! LinkedIn
Yammer All Conversations

Service ServiceRelay

U hebt een service gemaakt.

Om deze service te testen moet u een client maken en daarmee de service aanroepen. U kunt dit vanuit de opdrachtregel doen met het hulpprogramma svcutil.exe. Gebruik hiervoor de syntaxis:

```
svcutil.exe http://cslt-1113-01:12323/VOWS/ServiceRelay?wsdl
```

U kunt de servicebeschrijving ook als één bestand openen:

```
http://cslt-1113-01:12323/VOWS/ServiceRelay?singleWsdl
```

Hiermee wordt een configuratiebestand gegenereerd en een codebestand dat de clientklasse bevat. Voeg de twee bestanden toe aan de clienttoepassing en gebruik de gegenereerde clientklasse om de service aan te roepen. Bijvoorbeeld:

C#

```
class Test
{
    static void Main()
    {
        ServiceRelayClient client = new ServiceRelayClient();

        // Gebruik de variabele 'client' om bewerkingen voor de service aan te roepen.

        // Sluit de client altijd af.
        client.Close();
    }
}
```

Visual Basic

```
Class Test
    Shared Sub Main()
        Dim client As ServiceRelayClient = New ServiceRelayClient()
        ' Gebruik de variabele 'client' om bewerkingen voor de service aan te roepen.

        ' Sluit de client altijd af.
        client.Close()
    End Sub
End Class
```

If this page does not appear the **VOServiceRelay** service should be restarted. This can be done through **Windows / Task Manager / Services**.

Data Exchange Utility

Data Exchange Utility is a small application that should be mainly used to verify the proper functioning of VOWS XML services by calling all the operations the web services provide: export VPS information, project data and import project data.

The following sections will show you how to do a manual data export, export and import into an empty project and export VPS information. The data exchange utility tool can be found in the following folder: C:\Program Files\Vico Software\VOWS\Data Exchange Utility.

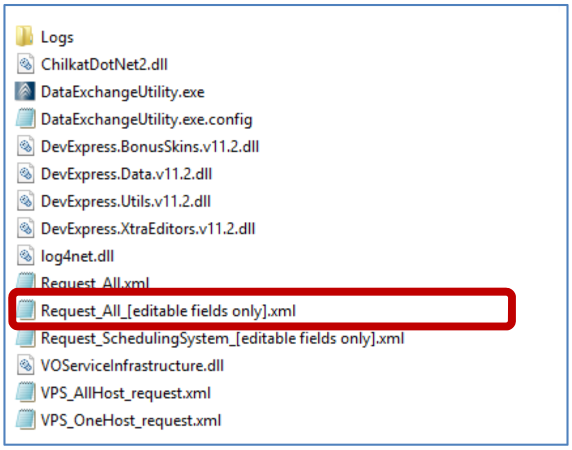
Manual data export

Go to the folder: *C:\Program Files\Vico Software\VOWS\Data Exchange Utility*, as seen here are some request XML template files that can be used for data exporting.

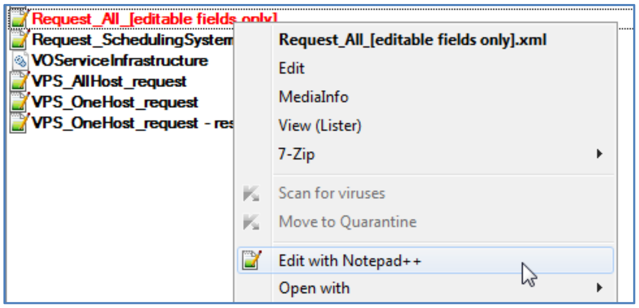
Note: It is important that the user has administrator privileges on this folder to perform manual data exchange!

Note: A text editor is needed to view and edit the XML templates and to view the response XMLs. The default Notepad can be used but the free **Notepad++** together with the **XML Tools** plugin is highly recommended – these can be of great help for coloring and formatting the XML.

For this example we'll use ***Request_All_[editable fields only].xml*** template. Select *Request_All_[editable fields only].xml* file and choose **Edit with Notepad++** or **Edit.**

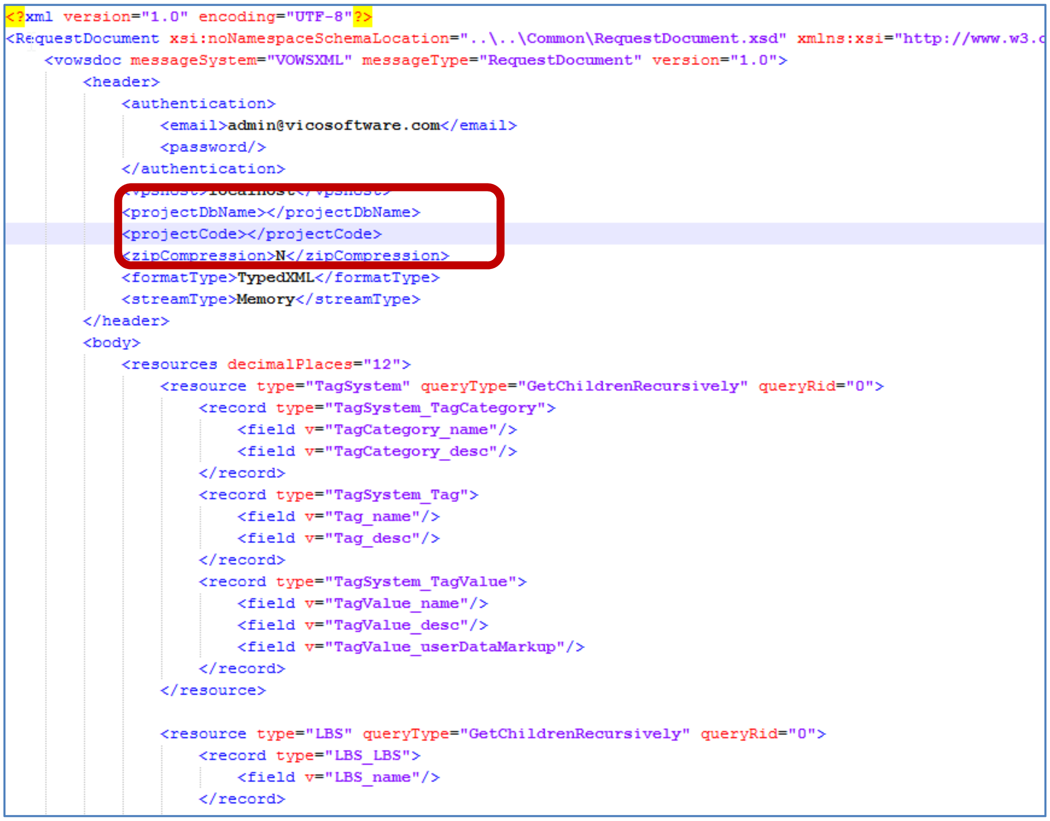


Logs
ChilkatDotNet2.dll
DataExchangeUtility.exe
DataExchangeUtility.exe.config
DevExpress.BonusSkins.v11.2.dll
DevExpress.Data.v11.2.dll
DevExpress.Utils.v11.2.dll
DevExpress.XtraEditors.v11.2.dll
log4net.dll
Request_All.xml
Request_All_[editable fields only].xml
Request_SchedulingSystem_[editable fields only].xml
VOServiceInfrastructure.dll
VPS_AllHost_request.xml
VPS_OneHost_request.xml



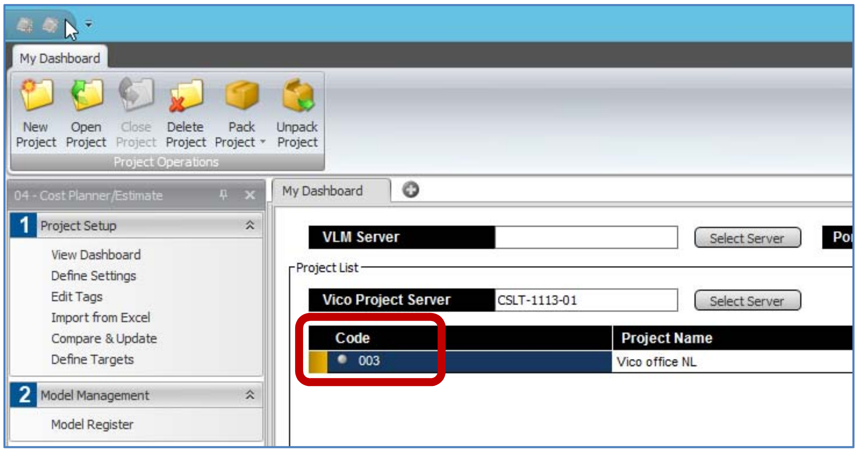
In order to export data from a project the VPS host and the project database name or project code must be specified in the request XML document – see below the highlighted fields that have to be modified. VPS host is set to localhost by default. If you want to export data from another VPS from the network replace localhost with the host name of the specific VPS.

It is more convenient to use the project code in the request XML then the project database name, as project code is available in *Vico Office \ Dashboard* view, therefore we will use this in the following examples. Project code can be filled between the >< symbols of projectCode element.



```
<?xml version="1.0" encoding="UTF-8"?>
<RequestDocument xsi:noNamespaceSchemaLocation="..\..\Common\RequestDocument.xsd" xmlns:xsi="http://www.w3.c
    <vowsdoc messageSystem="VOWSXML" messageType="RequestDocument" version="1.0">
        <header>
            <authentication>
                <email>admin@vicosoftware.com</email>
                <password/>
            </authentication>
            <vpsHost>localhost</vpsHost>
            <projectDbName></projectDbName>
            <projectCode></projectCode>
            <zipCompression>N</zipCompression>
            <formatType>TypedXML</formatType>
            <streamType>Memory</streamType>
        </header>
        <body>
            <resources decimalPlaces="12">
                <resource type="TagSystem" queryType="GetChildrenRecursively" queryRid="0">
                    <record type="TagSystem_TagCategory">
                        <field v="TagCategory_name"/>
                        <field v="TagCategory_desc"/>
                    </record>
                    <record type="TagSystem_Tag">
                        <field v="Tag_name"/>
                        <field v="Tag_desc"/>
                    </record>
                    <record type="TagSystem_TagValue">
                        <field v="TagValue_name"/>
                        <field v="TagValue_desc"/>
                        <field v="TagValue_userDataMarkup"/>
                    </record>
                </resource>

                <resource type="LBS" queryType="GetChildrenRecursively" queryRid="0">
                    <record type="LBS_LBS">
                        <field v="LBS_name"/>
                    </record>
```

Project code in Vico Office \ Dashboard view is highlighted below:

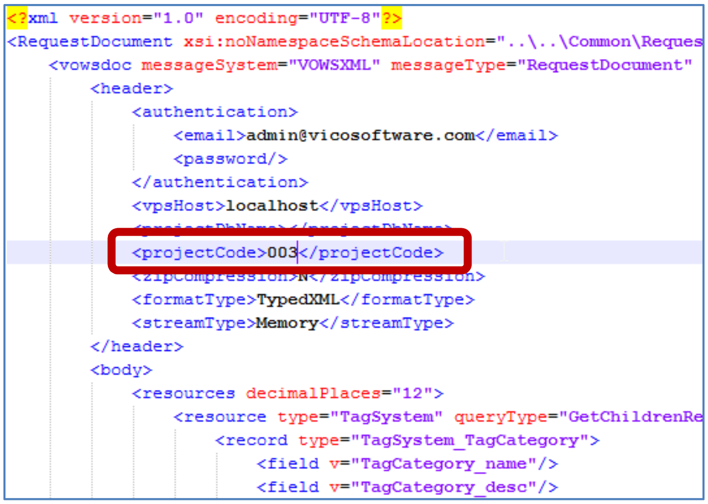


VLM Server

Vico Project Server CSLT-1113-01

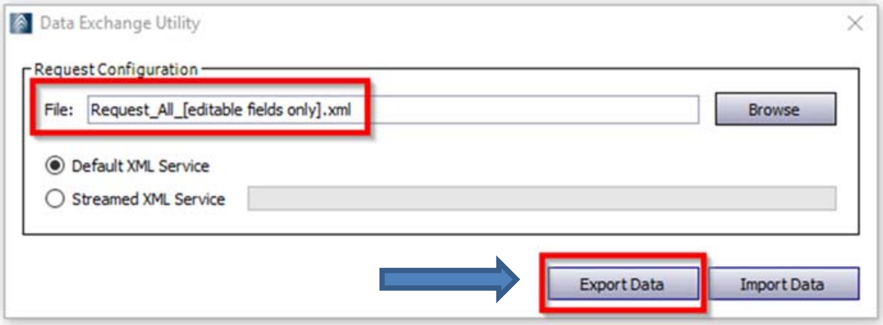
Code	Project Name
003	Vico office NL

Save the file after modifying the projectCode element.



```
<?xml version="1.0" encoding="UTF-8"?>
<RequestDocument xsi:noNamespaceSchemaLocation="..\..\Common\Reques
    <vowsdoc messageSystem="VOWSXML" messageType="RequestDocument"
        <header>
            <authentication>
                <email>admin@vicosoftware.com</email>
                <password/>
            </authentication>
            <vpsHost>localhost</vpsHost>
            <projectDbName></projectDbName>
            <projectCode>003</projectCode>
            <zipCompression>N</zipCompression>
            <formatType>TypedXML</formatType>
            <streamType>Memory</streamType>
        </header>
        <body>
            <resources decimalPlaces="12">
                <resource type="TagSystem" queryType="GetChildrenRe
                    <record type="TagSystem_TagCategory">
                        <field v="TagCategory_name"/>
                        <field v="TagCategory_desc"/>
```

Start the Data Exchange Utility program and copy the request file name (with extension) to the File field as shown below:



Data Exchange Utility

Request Configuration

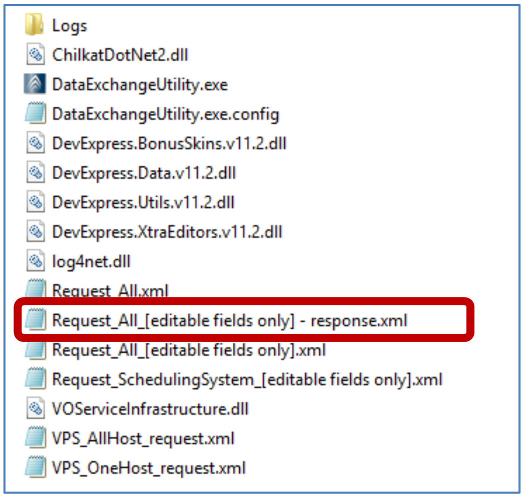
File: Request_All_[editable fields only].xml

● Default XML Service

○ Streamed XML Service

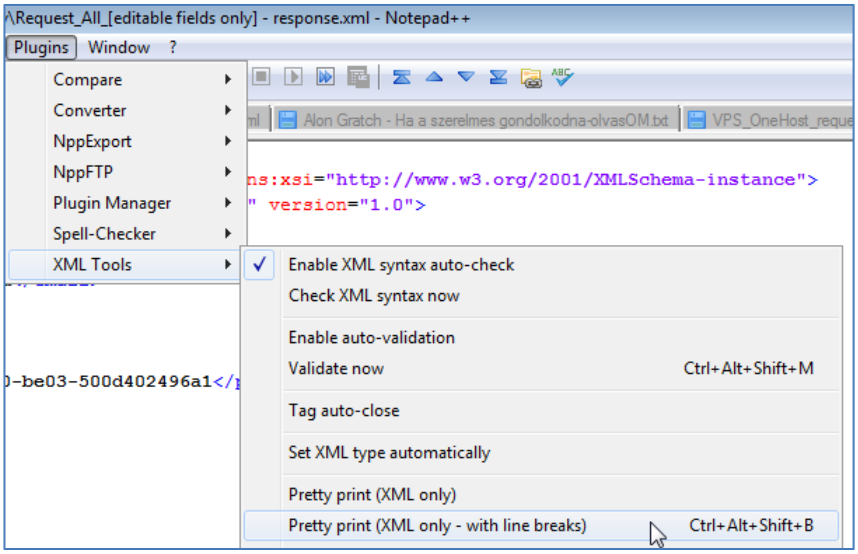
Click on the Export Data button; a response XML document will be created with all requested information. Depending on the project size and requested information the export process can take up to several minutes.

Open the response XML file (preferably with Notepad++) to verify that the data was exported correctly.



Logs
ChilkatDotNet2.dll
DataExchangeUtility.exe
DataExchangeUtility.exe.config
DevExpress.BonusSkins.v11.2.dll
DevExpress.Data.v11.2.dll
DevExpress.Utils.v11.2.dll
DevExpress.XtraEditors.v11.2.dll
log4net.dll
Request_All.xml
Request_All_[editable fields only] - response.xml
Request_All_[editable fields only].xml
Request_SchedulingSystem_[editable fields only].xml
VOServiceInfrastructure.dll
VPS_AllHost_request.xml
VPS_OneHost_request.xml

Note for **Notepad++:** In order to enhance XML coloring and formatting use the **XML Tools** plugin. This can be installed through **Plugins \ Plugin Manager**. Once installed click on **Pretty print (XML only – with line breaks)** to format the document (have the XML elements hierarchically aligned).

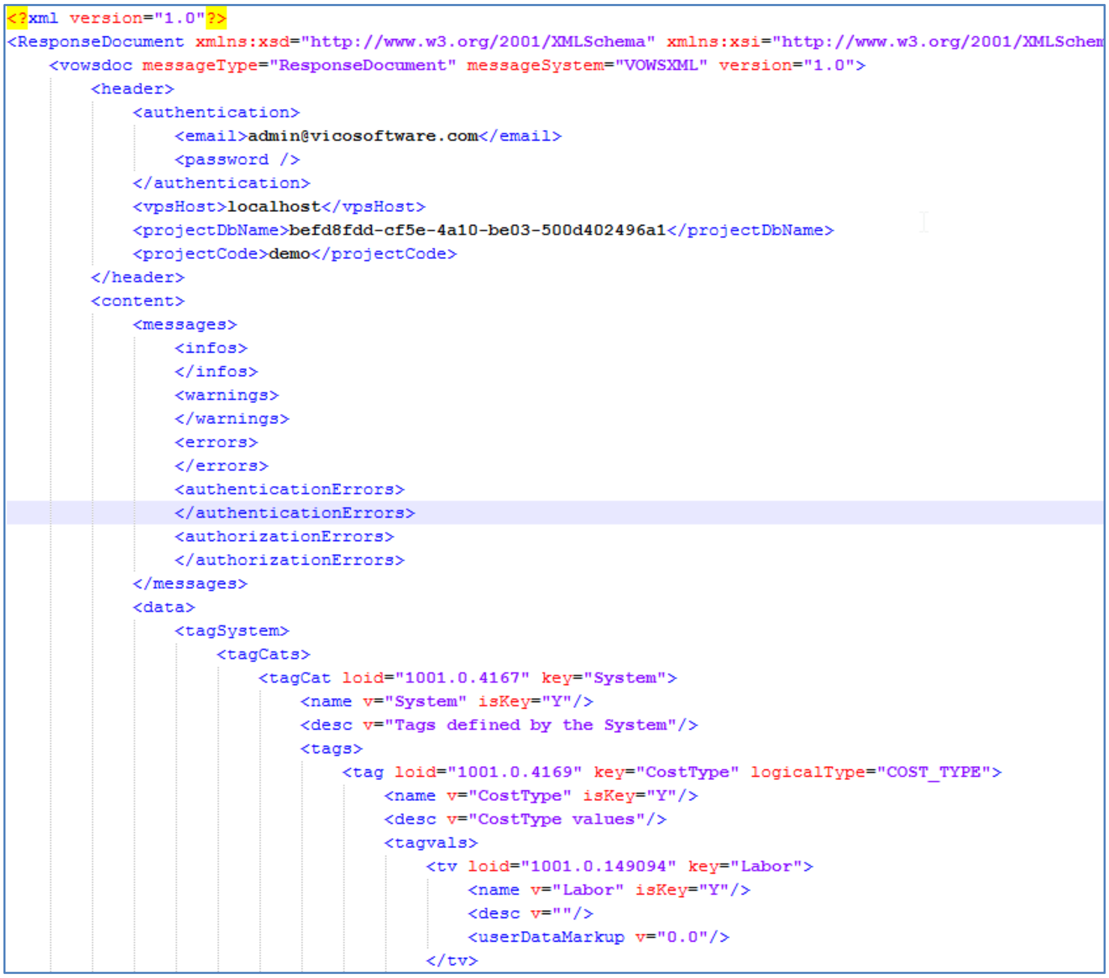


\Request_All_[editable fields only] - response.xml - Notepad++

```
ns:xsi="http://www.w3.org/2001/XMLSchema-instance">
" version="1.0">
)-be03-500d402496a1</
```

Plugins
Window
?
Compare
Converter
NppExport
NppFTP
Plugin Manager
Spell-Checker
XML Tools
Alon Gratch - Ha a szerelmes gondolkodna-olvasOM.txt
VPS_OneHost_reque
✓ Enable XML syntax auto-check
Check XML syntax now
Enable auto-validation
Validate now Ctrl+Alt+Shift+M
Tag auto-close
Set XML type automatically
Pretty print (XML only)
Pretty print (XML only - with line breaks) Ctrl+Alt+Shift+B

The correct export should have no warning or error messages, as seen below.

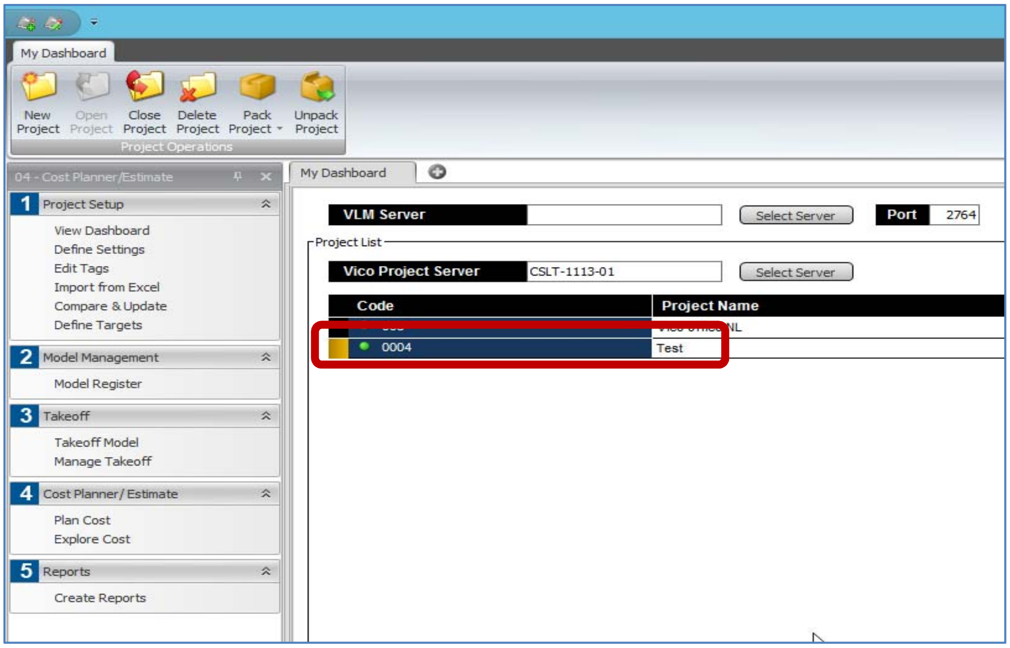


```
<?xml version="1.0"?>
<ResponseDocument xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchem
    <vowsdoc messageType="ResponseDocument" messageSystem="VOWSXML" version="1.0">
        <header>
            <authentication>
                <email>admin@vicosoftware.com</email>
                <password />
            </authentication>
            <vpsHost>localhost</vpsHost>
            <projectDbName>befd8fdd-cf5e-4a10-be03-500d402496a1</projectDbName>
            <projectCode>demo</projectCode>
        </header>
        <content>
            <messages>
                <infos>
                </infos>
                <warnings>
                </warnings>
                <errors>
                </errors>
                <authenticationErrors>
                </authenticationErrors>
                <authorizationErrors>
                </authorizationErrors>
            </messages>
            <data>
                <tagSystem>
                    <tagCats>
                        <tagCat loid="1001.0.4167" key="System">
                            <name v="System" isKey="Y"/>
                            <desc v="Tags defined by the System"/>
                            <tags>
                                <tag loid="1001.0.4169" key="CostType" logicalType="COST_TYPE">
                                    <name v="CostType" isKey="Y"/>
                                    <desc v="CostType values"/>
                                    <tagvals>
                                        <tv loid="1001.0.149094" key="Labor">
                                            <name v="Labor" isKey="Y"/>
                                            <desc v=""/>
                                            <userDataMarkup v="0.0"/>
                                        </tv>
```

Manual data export and import to another project

The project used in the previous example, **003** Vico office NL, is a fully developed project with a budget, locations, schedule etc. We export this information and import it to an empty project, **0004** Test project. The **Request_All_[*editable* fields only].xml** template will be used again to export project data. The file can be found in the folder: *C:*\Program Files\Vico Software\VOWS\Data Exchange Utility.

Create a new blank project in Vico Office and give it the code **0004** and name it **Test**.



VLM Server

Port 2764

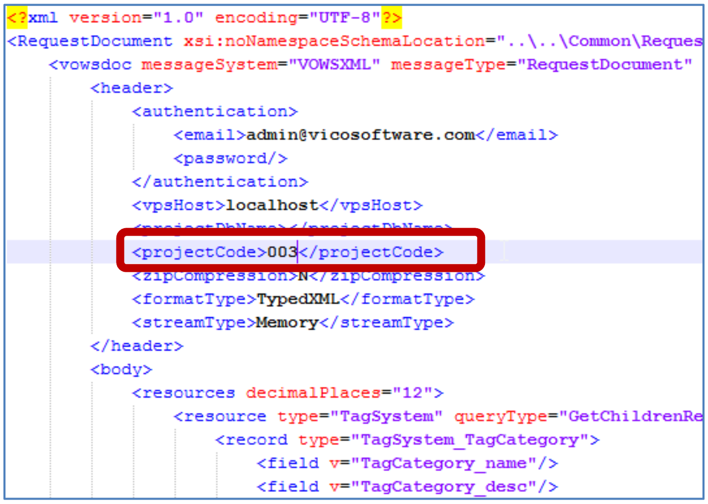
Project List

Vico Project Server CSLT-1113-01

Code	Project Name
003	Vico Office NL
0004	Test

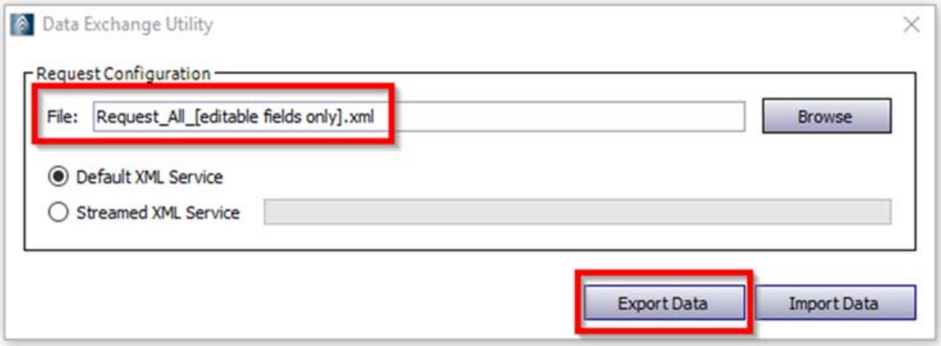
Step 1: (Export)

Edit the **Request_All_[*editable* fields only].xml** file and enter the source VPS host and project code.



```
<?xml version="1.0" encoding="UTF-8"?>
<RequestDocument xsi:noNamespaceSchemaLocation="..\..\Common\Reques
    <vowsdoc messageSystem="VOWSXML" messageType="RequestDocument"
        <header>
            <authentication>
                <email>admin@vicosoftware.com</email>
                <password/>
            </authentication>
            <vpsHost>localhost</vpsHost>
            <projectDbName></projectDbName>
            <projectCode>003</projectCode>
            <zipCompression>N</zipCompression>
            <formatType>TypedXML</formatType>
            <streamType>Memory</streamType>
        </header>
        <body>
            <resources decimalPlaces="12">
                <resource type="TagSystem" queryType="GetChildrenRe
                    <record type="TagSystem_TagCategory">
                        <field v="TagCategory_name"/>
                        <field v="TagCategory_desc"/>
```

Start the Data Exchange Utility program and copy the request file name, Request_All_[*editable* fields only].xml**,** to the File field as shown below:

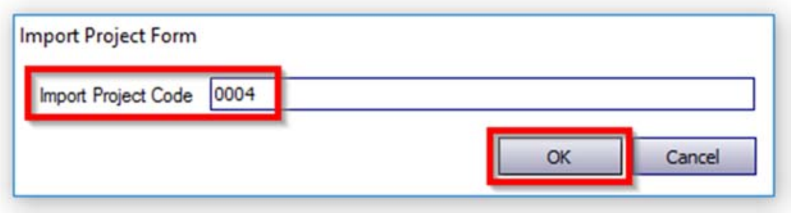


File: Request_All_[editable fields only].xml

● Default XML Service

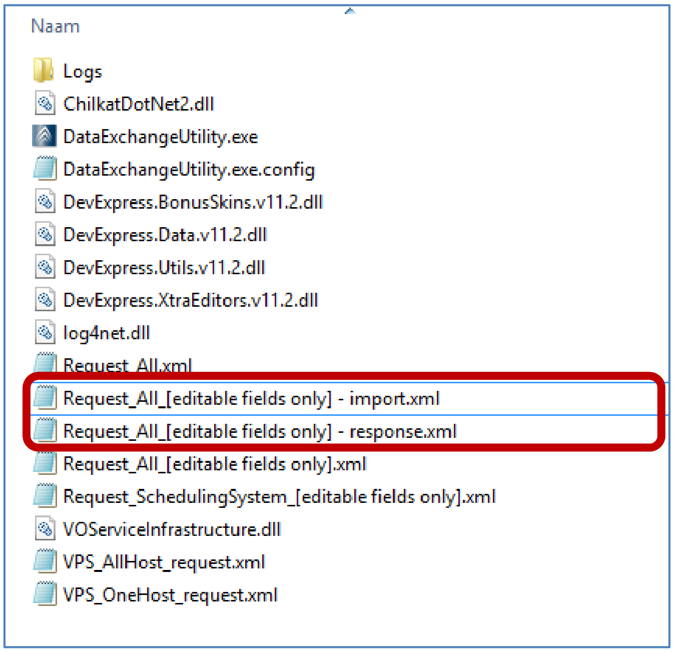
○ Streamed XML Service

Right click the Export Data button. In the dialog that pops-up enter the target (import) project code. Click the OK button to start the data export.



Import Project Code: 0004

Two documents are created through this export process: a response XML document with all requested information and a transmit XML document that can be used to import data in the specified project.

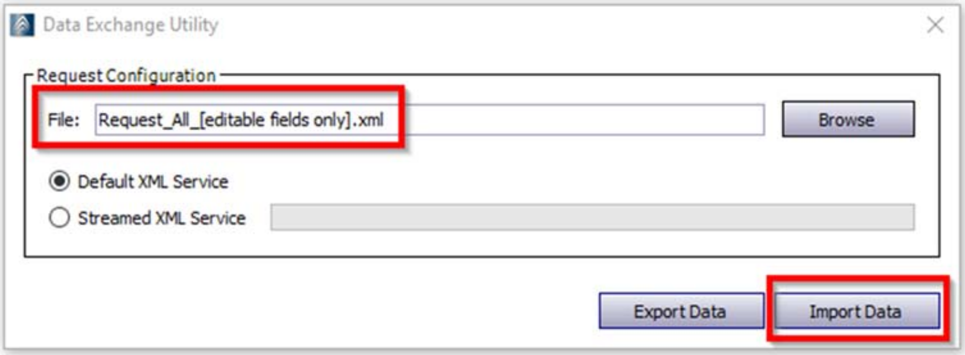


Naam

- Logs
- ChilkatDotNet2.dll
- DataExchangeUtility.exe
- DataExchangeUtility.exe.config
- DevExpress.BonusSkins.v11.2.dll
- DevExpress.Data.v11.2.dll
- DevExpress.Utils.v11.2.dll
- DevExpress.XtraEditors.v11.2.dll
- log4net.dll
- Request_All.xml
- Request_All_[editable fields only] - import.xml
- Request_All_[editable fields only] - response.xml
- Request_All_[editable fields only].xml
- Request_SchedulingSystem_[editable fields only].xml
- VOServiceInfrastructure.dll
- VPS_AllHost_request.xml
- VPS_OneHost_request.xml

Step 2: (Import)

Copy the name of the generated **Request_All_[*editable* fields only] - import.xml** file to the File field as shown below:



Data Exchange Utility

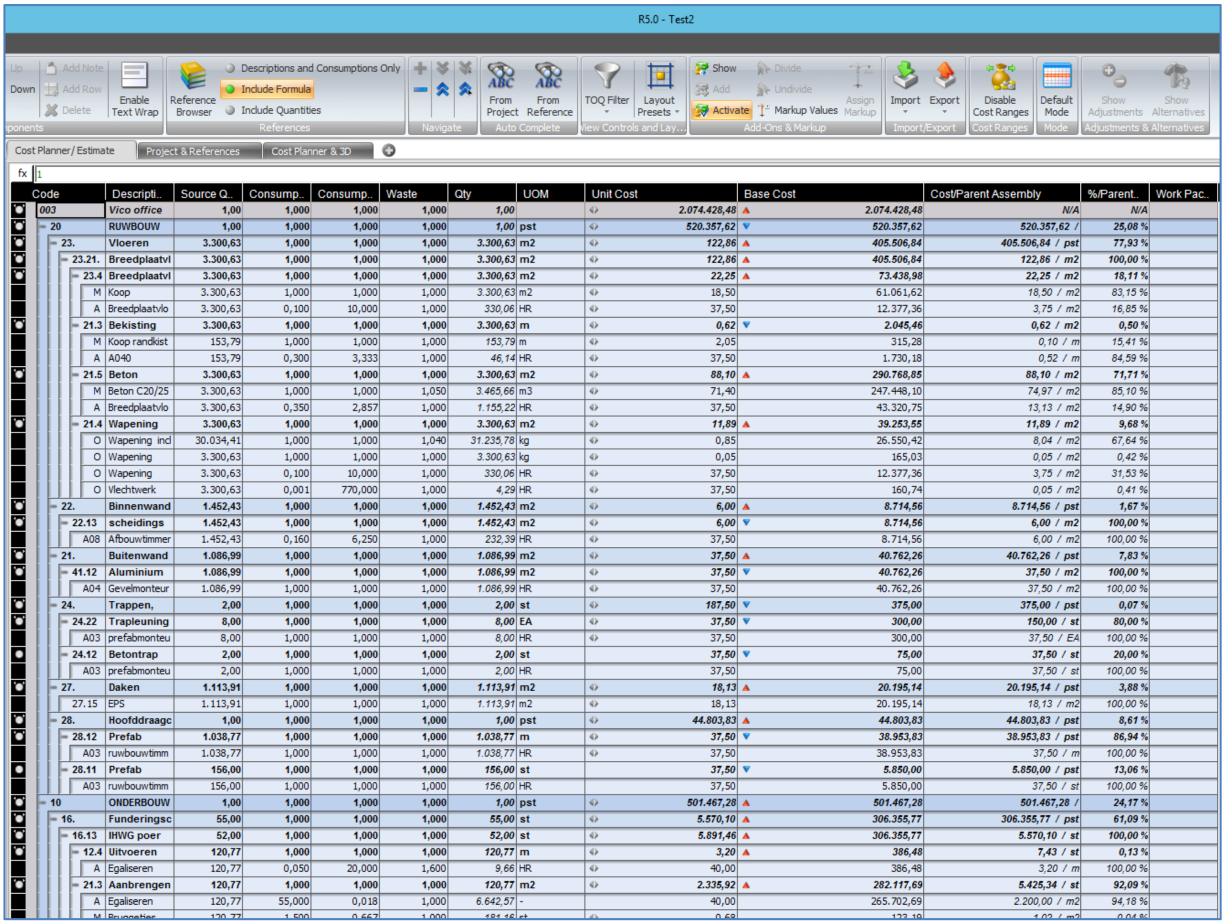
Request Configuration

File: Request_All_[editable fields only].xml

● Default XML Service
○ Streamed XML Service

Click Import Data button to start the import process. The import process is much slower than the export, be prepared to wait up to an hour for very large (>200MB) import files. The import file generated for this test (~300KB) was imported in less than 10 seconds.

All the data is now imported into the **0004** test project; compare the data against the original project.



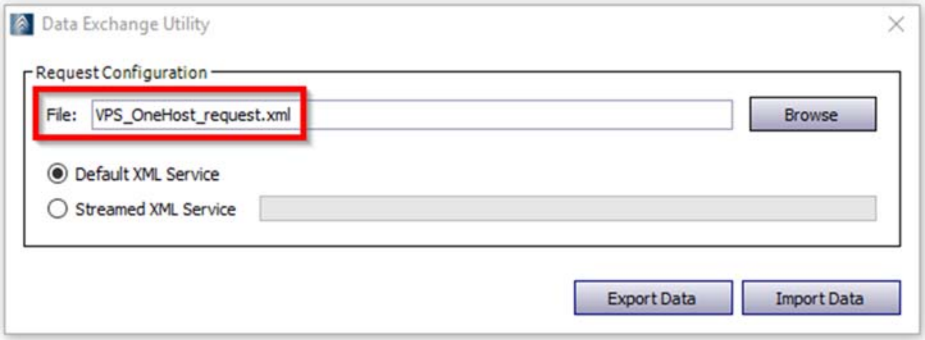
fx 1

Code	Descripti..	Source Q..	Consump..	Consump..	Waste	Qty	UOM	Unit Cost	Base Cost	Cost/Parent Assembly	%/Parent..	Work Pac..
003	Vico office	1,00	1,000	1,000	1,000	1,00		2.074.428,48	▲ 2.074.428,48	N/A	N/A	
20	RUWBOUW	1,00	1,000	1,000	1,000	1,00	pst	520.357,62	▼ 520.357,62	520.357,62 /	25,08 %	
23.	Vloeren	3.300,63	1,000	1,000	1,000	3.300,63	m2	122,86	▲ 405.506,84	405.506,84 / pst	77,93 %	
23.21.	Breedplaatvl	3.300,63	1,000	1,000	1,000	3.300,63	m2	122,86	▲ 405.506,84	122,86 / m2	100,00 %	
23.4	Breedplaatvl	3.300,63	1,000	1,000	1,000	3.300,63	m2	22,25	▲ 73.438,98	22,25 / m2	18,11 %	
M	Koop	3.300,63	1,000	1,000	1,000	3.300,63	m2	18,50	61.061,62	18,50 / m2	83,15 %	
A	Breedplaatvlo	3.300,63	0,100	10,000	1,000	330,06	HR	37,50	12.377,36	3,75 / m2	16,85 %	
21.3	Bekisting	3.300,63	1,000	1,000	1,000	3.300,63	m	0,62	▼ 2.045,46	0,62 / m2	0,50 %	
M	Koop randkist	153,79	1,000	1,000	1,000	153,79	m	2,05	315,28	0,10 / m	15,41 %	
A	A040	153,79	0,300	3,333	1,000	46,14	HR	37,50	1.730,18	0,52 / m	84,59 %	
21.5	Beton	3.300,63	1,000	1,000	1,000	3.300,63	m2	88,10	▲ 290.768,85	88,10 / m2	71,71 %	
M	Beton C20/25	3.300,63	1,000	1,000	1,050	3.465,66	m3	71,40	247.448,10	74,97 / m2	85,10 %	
A	Breedplaatvlo	3.300,63	0,350	2,857	1,000	1.155,22	HR	37,50	43.320,75	13,13 / m2	14,90 %	
21.4	Wapening	3.300,63	1,000	1,000	1,000	3.300,63	m2	11,89	▲ 39.253,55	11,89 / m2	9,68 %	
O	Wapening incl	30.034,41	1,000	1,000	1,040	31.235,78	kg	0,85	26.550,42	8,04 / m2	67,64 %	
O	Wapening	3.300,63	1,000	1,000	1,000	3.300,63	kg	0,05	165,03	0,05 / m2	0,42 %	
O	Wapening	3.300,63	0,100	10,000	1,000	330,06	HR	37,50	12.377,36	3,75 / m2	31,53 %	
O	Vlechtwerk	3.300,63	0,001	770,000	1,000	4,29	HR	37,50	160,74	0,05 / m2	0,41 %	
22.	Binnenwand	1.452,43	1,000	1,000	1,000	1.452,43	m2	6,00	▲ 8.714,56	8.714,56 / pst	1,67 %	
22.13	scheidings	1.452,43	1,000	1,000	1,000	1.452,43	m2	6,00	▼ 8.714,56	6,00 / m2	100,00 %	
A08	Afbouwtimmer	1.452,43	0,160	6,250	1,000	232,39	HR	37,50	8.714,56	6,00 / m2	100,00 %	
21.	Buitenwand	1.086,99	1,000	1,000	1,000	1.086,99	m2	37,50	▲ 40.762,26	40.762,26 / pst	7,83 %	
41.12	Aluminium	1.086,99	1,000	1,000	1,000	1.086,99	m2	37,50	▼ 40.762,26	37,50 / m2	100,00 %	
A04	Gevelmonteur	1.086,99	1,000	1,000	1,000	1.086,99	HR	37,50	40.762,26	37,50 / m2	100,00 %	
24.	Trappen,	2,00	1,000	1,000	1,000	2,00	st	187,50	▼ 375,00	375,00 / pst	0,07 %	
24.22	Trapleuning	8,00	1,000	1,000	1,000	8,00	EA	37,50	▼ 300,00	150,00 / st	80,00 %	
A03	prefabmonteu	8,00	1,000	1,000	1,000	8,00	HR	37,50	300,00	37,50 / EA	100,00 %	
24.12	Betontrap	2,00	1,000	1,000	1,000	2,00	st	37,50	▼ 75,00	37,50 / st	20,00 %	
A03	prefabmonteu	2,00	1,000	1,000	1,000	2,00	HR	37,50	75,00	37,50 / st	100,00 %	
27.	Daken	1.113,91	1,000	1,000	1,000	1.113,91	m2	18,13	▲ 20.195,14	20.195,14 / pst	3,88 %	
27.15	EPS	1.113,91	1,000	1,000	1,000	1.113,91	m2	18,13	20.195,14	18,13 / m2	100,00 %	
28.	Hoofddraagc	1,00	1,000	1,000	1,000	1,00	pst	44.803,83	▲ 44.803,83	44.803,83 / pst	8,61 %	
28.12	Prefab	1.038,77	1,000	1,000	1,000	1.038,77	m	37,50	▼ 38.953,83	38.953,83 / pst	86,94 %	
A03	ruwbouwtimm	1.038,77	1,000	1,000	1,000	1.038,77	HR	37,50	38.953,83	37,50 / m	100,00 %	
28.11	Prefab	156,00	1,000	1,000	1,000	156,00	st	37,50	▼ 5.850,00	5.850,00 / pst	13,06 %	
A03	ruwbouwtimm	156,00	1,000	1,000	1,000	156,00	HR	37,50	5.850,00	37,50 / st	100,00 %	
10	ONDERBOUW	1,00	1,000	1,000	1,000	1,00	pst	501.467,28	▲ 501.467,28	501.467,28 /	24,17 %	
16.	Funderingsc	55,00	1,000	1,000	1,000	55,00	st	5.570,10	▲ 306.355,77	306.355,77 / pst	61,09 %	
16.13	IHWG poer	52,00	1,000	1,000	1,000	52,00	st	5.891,46	▲ 306.355,77	5.570,10 / st	100,00 %	
12.4	Uitvoeren	120,77	1,000	1,000	1,000	120,77	m	3,20	▲ 386,48	7,43 / st	0,13 %	
A	Egaliseren	120,77	0,050	20,000	1,600	9,66	HR	40,00	386,48	3,20 / m	100,00 %	
21.3	Aanbrengen	120,77	1,000	1,000	1,000	120,77	m2	2.335,92	▲ 282.117,69	5.425,34 / st	92,09 %	
A	Egaliseren	120,77	55,000	0,018	1,000	6.642,57	-	40,00	265.702,69	2.200,00 / m2	94,18 %	
M	Bruggetjes	120,77	1,500	0,667	1,000	181,16	st	0,68	123,19	1,02 / m2	0,04 %	

Manual export of VPS host information

There are two request templates for getting VPS host information: **VPS_AllHost_request.xml** and **VPS_OneHost_request.xml**. The first one will scan for VPS information all the machines from the LAN, therefore for large LANs this process can take several minutes. The second one, used in this example, will export VPS information from the local machine only.

Start the Data Exchange Utility program and copy the **VPS_OneHost_request.xml** file name to the File field as shown below:

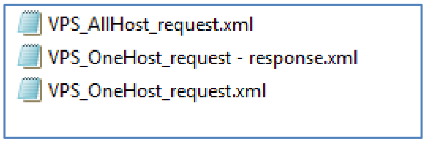


File: VPS_OneHost_request.xml

● Default XML Service

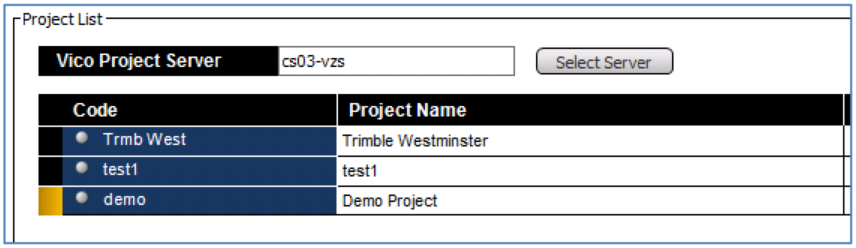
○ Streamed XML Service

Click the Export Data button. A response file containing project information for the given VPS host is generated.



VPS_AllHost_request.xml
VPS_OneHost_request - response.xml
VPS_OneHost_request.xml

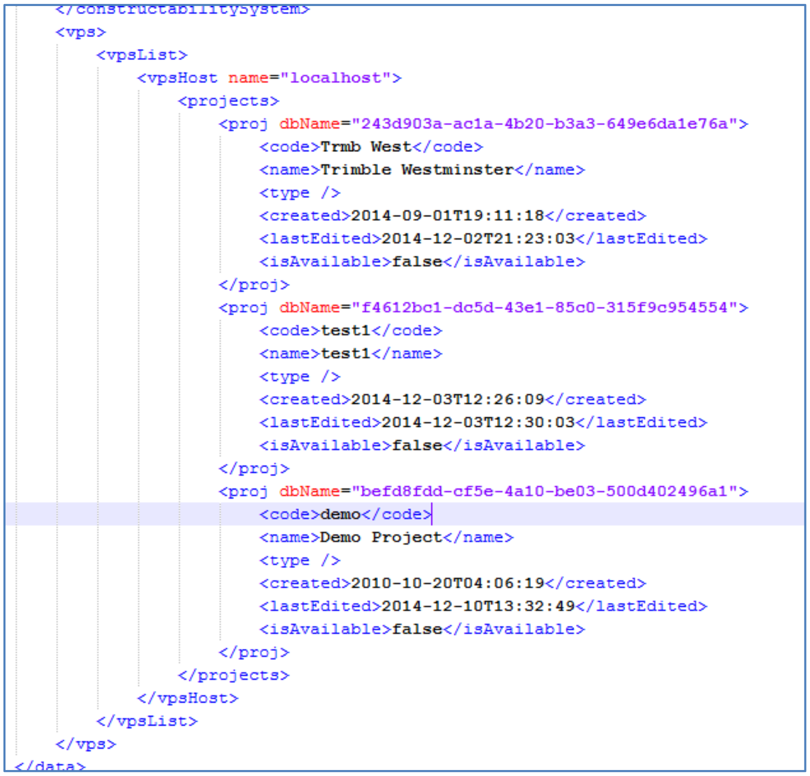
Vico Office \ Dashboard view shows all the projects from the local machine.



Vico Project Server: cs03-vzs

Code	Project Name
Trmb West	Trimble Westminster
test1	test1
demo	Demo Project

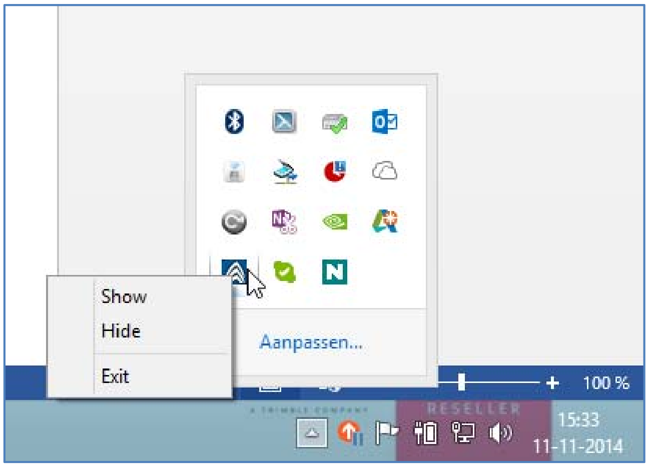
Open the generated ***VPS*_OneHosts_request – response.xml** file to view the results, see below an excerpt with the VPS information:

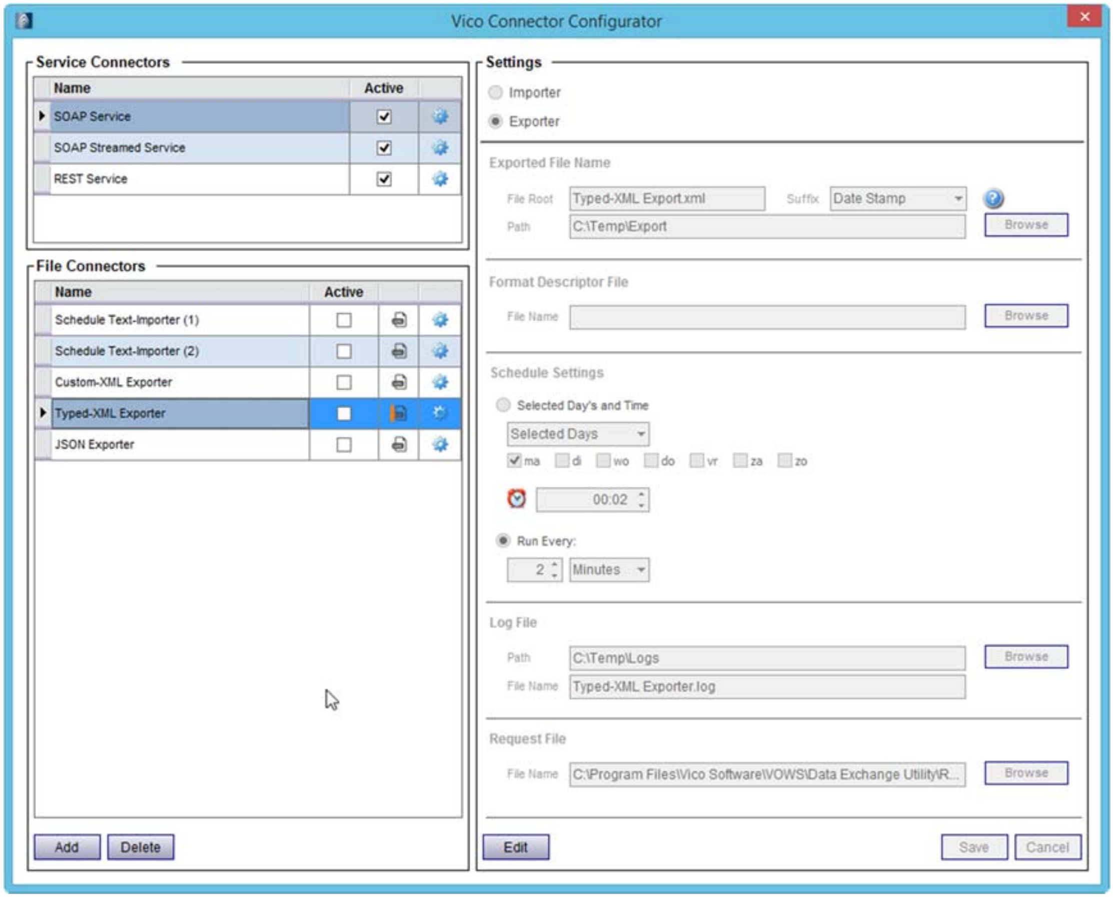


```
    </constructabilitySystem>
    <vps>
        <vpsList>
            <vpsHost name="localhost">
                <projects>
                    <proj dbName="243d903a-ac1a-4b20-b3a3-649e6da1e76a">
                        <code>Trmb West</code>
                        <name>Trimble Westminster</name>
                        <type />
                        <created>2014-09-01T19:11:18</created>
                        <lastEdited>2014-12-02T21:23:03</lastEdited>
                        <isAvailable>false</isAvailable>
                    </proj>
                    <proj dbName="f4612bc1-dc5d-43e1-85c0-315f9c954554">
                        <code>test1</code>
                        <name>test1</name>
                        <type />
                        <created>2014-12-03T12:26:09</created>
                        <lastEdited>2014-12-03T12:30:03</lastEdited>
                        <isAvailable>false</isAvailable>
                    </proj>
                    <proj dbName="befd8fdd-cf5e-4a10-be03-500d402496a1">
                        <code>demo</code>
                        <name>Demo Project</name>
                        <type />
                        <created>2010-10-20T04:06:19</created>
                        <lastEdited>2014-12-10T13:32:49</lastEdited>
                        <isAvailable>false</isAvailable>
                    </proj>
                </projects>
            </vpsHost>
        </vpsList>
    </vps>
</data>
```

Setting up a scheduled Export with Vico Connector Configurator

Start Vico Connector Configurator from taskbar icons by right-click on Vico Connector Configurator and choose Show. (See picture)





Vico Connector Configurator

Service Connectors

Name	Active
SOAP Service	☑
SOAP Streamed Service	☑
REST Service	☑

File Connectors

Name	Active
Schedule Text-Importer (1)	☐
Schedule Text-Importer (2)	☐
Custom-XML Exporter	☐
Typed-XML Exporter	☐
JSON Exporter	☐

Settings

○ Importer
● Exporter

Exported File Name

File Root: Typed-XML Export.xml
Suffix: Date Stamp
Path: C:\Temp\Export

Format Descriptor File

File Name:

Schedule Settings

○ Selected Day's and Time
Selected Days
☑ ma ☐ di ☐ wo ☐ do ☐ vr ☐ za ☐ zo
00:02
● Run Every:
2 Minutes

Log File

Path: C:\Temp\Logs
File Name: Typed-XML Exporter.log

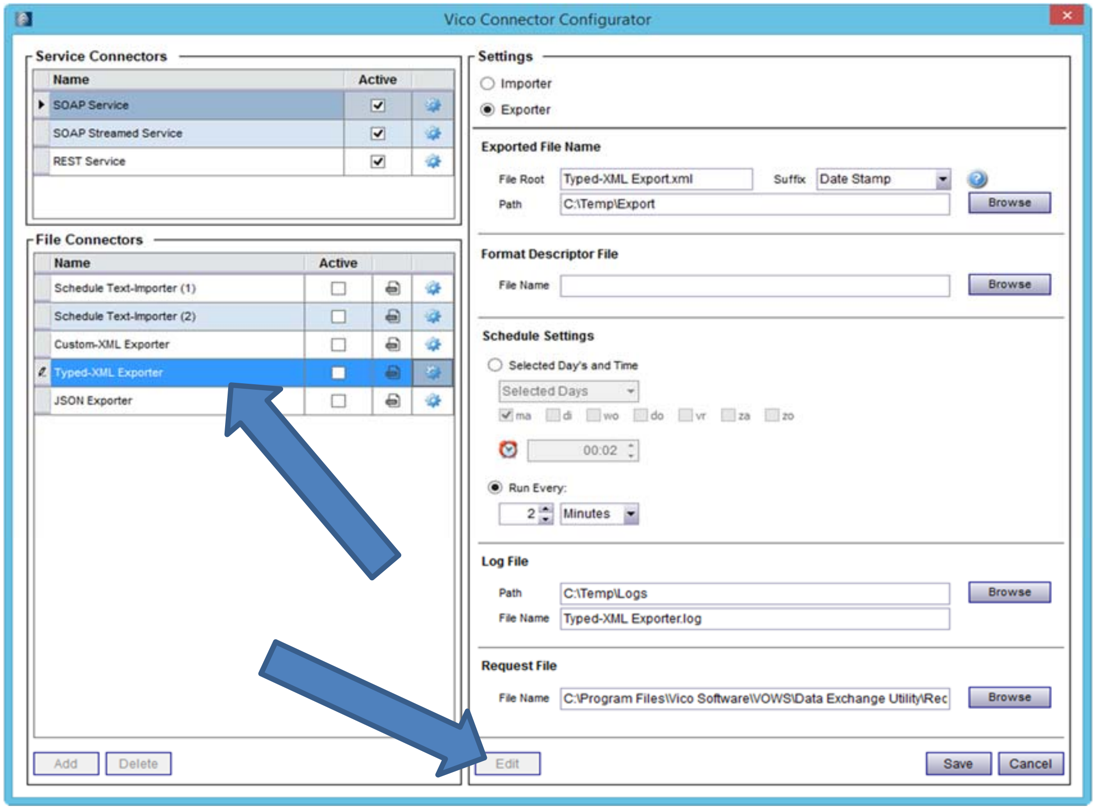
Request File

File Name: C:\Program Files\Vico Software\VOWS\Data Exchange Utility\R...

Note: Vico Connector Service and Vico Connector Configurator UI are explained in the Vico Connector Service & Configurator section

Scheduled exports using Typed XML Exporter

Select in the File Connectors window the **Typed XML Exporter** connector and click the Edit button.



Vico Connector Configurator

Service Connectors

Name	Active
SOAP Service	☑
SOAP Streamed Service	☑
REST Service	☑

File Connectors

Name	Active
Schedule Text-Importer (1)	□
Schedule Text-Importer (2)	□
Custom-XML Exporter	□
Typed-XML Exporter	□
JSON Exporter	□

Settings
○ Importer
● Exporter

Exported File Name
File Root: Typed-XML Export.xml Suffix: Date Stamp
Path: C:\Temp\Export

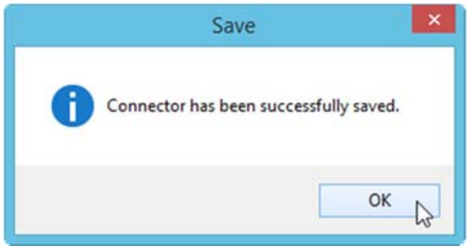
Format Descriptor File
File Name:

Schedule Settings
○ Selected Day's and Time
Selected Days
☑ ma □ di □ wo □ do □ vr □ za □ zo
00:02
● Run Every:
2 Minutes

Log File
Path: C:\Temp\Logs
File Name: Typed-XML Exporter.log

Request File
File Name: C:\Program Files\Vico Software\VOWS\Data Exchange Utility\Rec

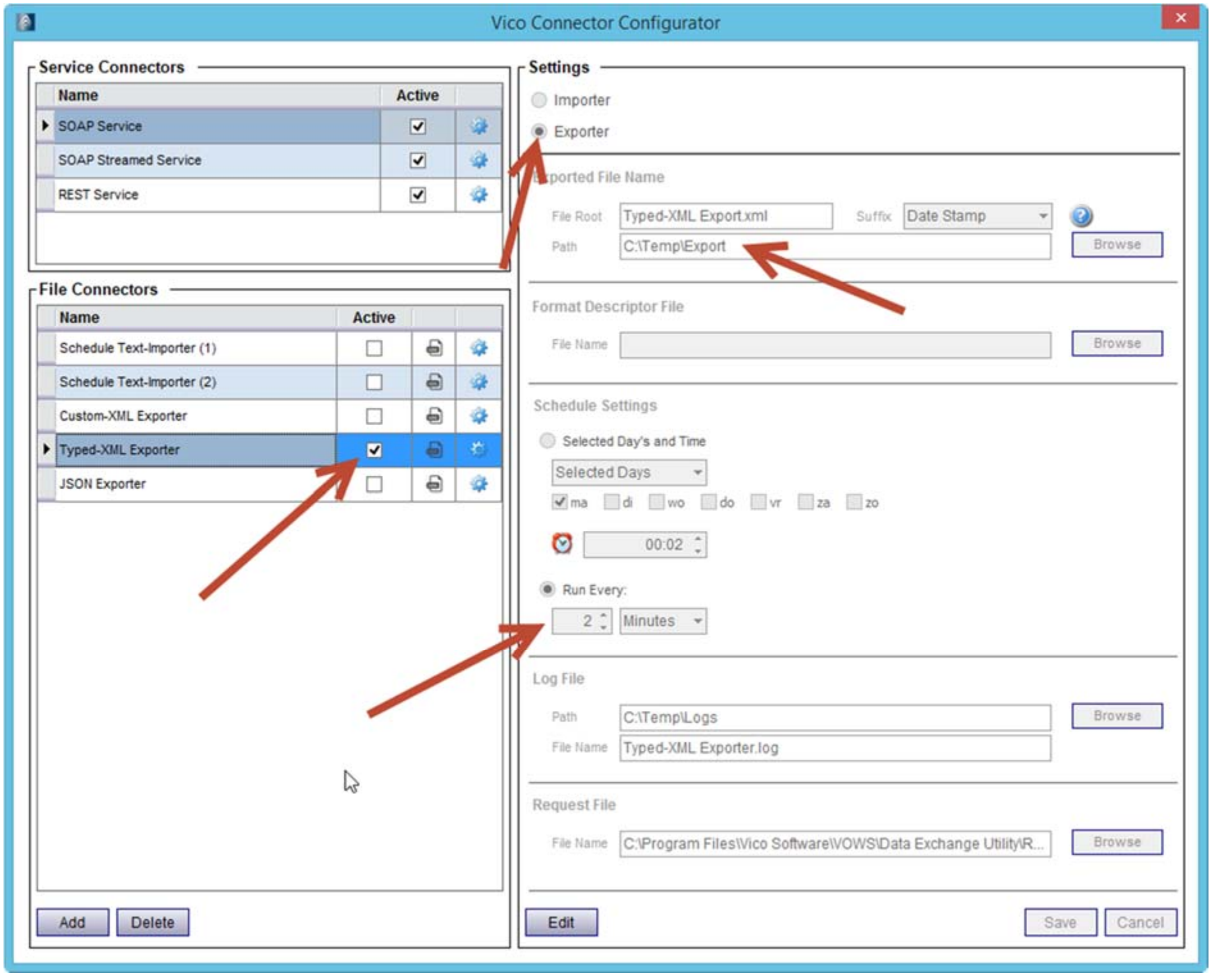
- Select the setting "Exporter"
- In the **Exported File Name** section set the Path where the exports are going to be saved (in this case C:\Temp\Export)
- The **Format Descriptor File** may remain empty – data will be saved in Data Exchange XML format
- Select "*Run Every*" in **Schedule Settings** and set it to be performed every two minutes
- In the **Log File** section set the *Path* where export logs are going to be saved
- Browse for a request XML file; here the **Request_All_ [*editable* fields only] .xml** file was selected. Note that VPS host and project code must be set in the request XML, see Step 1: (Exports)
- Check the Active checkbox for the Typed XML Exporter
- Saving the changes should display this message:



Save

Connector has been successfully saved.

After saving, Vico Connector Configurator user interface should look like this:



Vico Connector Configurator

Service Connectors

Name	Active
SOAP Service	☑
SOAP Streamed Service	☑
REST Service	☑

File Connectors

Name	Active
Schedule Text-Importer (1)	☐
Schedule Text-Importer (2)	☐
Custom-XML Exporter	☐
Typed-XML Exporter	☑
JSON Exporter	☐

Settings

○ Importer
● Exporter

Exported File Name
File Root: Typed-XML Export.xml Suffix: Date Stamp
Path: C:\Temp\Export

Format Descriptor File
File Name:

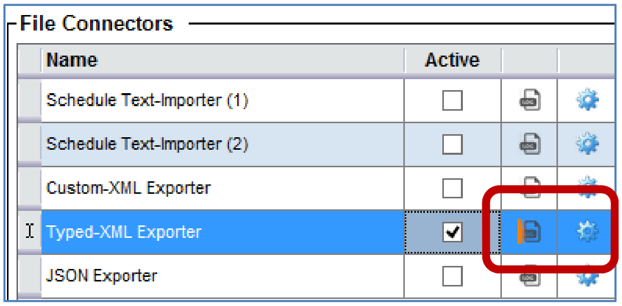
Schedule Settings
○ Selected Day's and Time
Selected Days
☑ ma ☐ di ☐ wo ☐ do ☐ vr ☐ za ☐ zo
00:02
● Run Every:
2 Minutes

Log File
Path: C:\Temp\Logs
File Name: Typed-XML Exporter.log

Request File
File Name: C:\Program Files\Vico Software\VOWS\Data Exchange Utility\R...

Verify in the export folder that a new export is generated every 2 minutes.

When a new file is generated a red vertical stripe is shown on the file icon of the connector.



File Connectors

Name	Active
Schedule Text-Importer (1)	☐
Schedule Text-Importer (2)	☐
Custom-XML Exporter	☐
Typed-XML Exporter	☑
JSON Exporter	☐