[Содержание](#Содержание)

# [How to automatically load flat files from an FTP server into SAP BW](#load_flat_files_from_FTP_into_BW)

# [How to create a bw datasource through XML file](#create_datasource_through_XML)

# [Configure a webservice datasource in #BW](#Configure_webservice_datasource)

# [Consuming a Web Service](#Consuming_Web_Service)

# [Create the XML data source](#create_XML_DataSource)

# [XML data loading to BW](#XML_data_loading_to_BW)

**Processing XML data in SAP HANA**

<https://blogs.sap.com/2017/08/14/processing-xml-data-in-sap-hana/>

# [Parsing XML within SAP ABAP](https://stackoverflow.com/questions/1765936/parsing-xml-within-sap-abap)

# <https://stackoverflow.com/questions/1765936/parsing-xml-within-sap-abap>

# [SAP BI](https://sap-bi-tai.blogspot.com/)

### XML-парсер

# <https://sap-bi-tai.blogspot.com/2015/07/xml.html>

## **Guide towards a simple conversion of an XML file to ABAP Internal table, using XML parsing**

# <http://saptechnical.com/Tutorials/ABAP/XML/Convert2.htm>

# Insert data into custom tables – Flat file loads to SAP HANA

# <http://teachmehana.com/loading-data-flat-file-sap-hana/>

# 

# How to create a bw datasource through XML file

# <https://answers.sap.com/questions/562450/how-to-create-a-bw-datasource-through-xml-file.html>

[Содержание](#Содержание)

# − How to create a bw datasource through XML file?

− Check if web-services option works for you -

* <https://blogs.sap.com/2014/01/17/configure-a-webservice-datasource-in-bw>/
* <https://help.sap.com/doc/saphelp_dm40/4.0/en-US/ab/0709407448c442e10000000a1550b0/content.htm?no_cache=true>

Web-service datasources works on Push-service mechanism - unlike other data sources in BW - *Web-services enabled datasource pushes the data from external system. BW don't pull the data in such case*.

Please go through following thread - <https://archive.sap.com/discussions/thread/1906140>

# − How to load data from a WebService to a BI7 DataSource?

A WebSerive-Datasource allows you to publish a WebService on the BW-System which waits for incomming data-deliveries.

So this whould not be the way I'm looking for since I'd like to pull data from an external webservice to the BW-system - triggered by the last one.

Are there other ways to pick-up the data from the external webservice and push them to the PSA?

I could imagine that there's some kind of module that consumes the external webservice and pushes it to the local webservice.

I found the expression proxy in that context repeatedly - but did not manage to fill that with life so far.

Could you please gimme some hints/ideas/comments about that topic?

− Here is how to consume WebService in ABAP.

<http://help.sap.com/saphelp_nw70/helpdata/EN/bf/d005244e9d1d4d92b2fe7935556b4c/frameset.htm>

According to this help document - you can call external WebService in forms of Method in ABAP program.

I think your idea can be achived by 2 steps.

1. Call external WebService sending a request message and recieving some data.
2. Call BW Datasource WebService sending data to be stored in BW as a request message.

# 

# Configure a webservice datasource in #BW

# <https://blogs.sap.com/2014/01/17/configure-a-webservice-datasource-in-bw/>

[Содержание](#Содержание)

# Introduction

The webservice datasource works with a push mechanicm allow the data delivery from the source system which is initiating the extraction process.

### **Configuring the webservice,**

which uses the standardized XML format is done in the administrator workbench.

To receive soap documents the SOAP service needs to be active in SICF only then data can be sent to bw through webservices.

# 

# Before the datasource can be configured a sources system of the type webservice needs to be created.

# 

# Maintain the necessary fields in the datasource and create an infopackage.

# 

# 

# On the web services navigator test the WSDL service interface which can be access by going through the the Netweaver Java Application Server.

# 

To retrieve the URL with the XML definition go through WSADMIN - the transaction is obsolete from SAP NetWeaver AS 7.00, SP14 - use SOAMANAGER.

Make sure that the port is setup properly in SAP Netweaver Administrator –> SOA tab –> SOA Middleware Global Settings –> HTTP Proxy tab.

Take the URL and provide it in the Web Service Navigator to test your webservice.  The webservice shows the defined fields provided in the definition of the datasource. After the import *manage PSA* will show the new created entry.

# 

# The webservice is working.

# 

# The XML-Schema defines the rules how the XML document has to be structured. The document used to exchange data needs to be well-formed and valid. Meaning it has to contain all rules according to W3C.

# Consuming a Web Service

[http://help.sap.com/saphelp\_nw70/helpdata/EN/bf/d005244e9d1d4d92b2fe7935556b4c/frameset.htm](http://help.sap.com/saphelp_nw70/helpdata/EN/bf/d005244e9d1d4d92b2fe7935556b4c/frameset.htm" \t "_blank)

[Содержание](#Содержание)

To consume a Web service - you can create a **proxy** for the Web service in the ABAP Workbench using a WSDL document as a basis. This can be done with just a few mouse clicks.

The role of the Web service client developer is provided for binding Web services.

The Web service client developer consumes the Web services in existing applications. He or she knows the specific requirements for the Web service and creates the client proxy. He or she then programs the client application.

## Prerequisite

You have configured the Web service runtime -refer to [Configuring the Web Service Runtime](https://help.sap.com/viewer/12aa7f056c531014aa5bca7aee037e55/7.0.37/en-US/47ab7d3b49f84848e10000000a421138.html).

## Procedure

1. Call the Object Navigator SE80 and create the proxy object - refer also to Generating a Proxy.
2. Create a logical port to define the runtime features of the Web service client proxy - refer also to [Configuring a Consumer Proxy](https://help.sap.com/viewer/12aa7f056c531014aa5bca7aee037e55/7.0.37/en-US/9ec7a3591dc74a679bbc9716354e42af.html).
3. Program your application - refer also to [Programming with Client and Server Proxies](https://help.sap.com/viewer/12aa7f056c531014aa5bca7aee037e55/7.0.37/en-US/c974246d8ad2447799063d39013e9a11.html).

# How to Load Transaction Data from Flat File in SAP BI/BW

# <https://www.guru99.com/how-to-load-transaction-data-from-flat-file.html>

# SAP BW Flat File Loading to InfoObject

# <https://erproof.com/bi/sap-bw-training/sap-bw-flat-file-loading-to-infoobject/>

# [How to read an XML file file in abap](https://stackoverflow.com/questions/9325581/how-to-read-an-xml-file-file-in-abap)

# <https://stackoverflow.com/questions/9325581/how-to-read-an-xml-file-file-in-abap>

# Load XML into SAP BW 7.3

# <https://answers.sap.com/questions/10709406/load-xml-into-sap-bw-73.html>

# 

# How to automatically load flat files from an FTP server into SAP BW

# <https://www.element61.be/en/resource/how-automatically-load-flat-files-ftp-server-sap-bw>

[Содержание](#Содержание)

# The process of loading flat files into SAP BW can be fully automated by using an FTP server on which the files are being stored. The reason for an FTP server - rather than granting the right to put files on the SAP BW server - is that we can keep control of what files end up (оказываются) on the SAP BW machine and we minimize the risk of running out of disk space or security risks.

# We will also show how a flat file on the SAP BW application server can be loaded into the PSA - Persistent Staging Area.

### **Flat files on an FTP server**

# To access an FTP server via your Internet Browser you will need the host name of the server. Its address will be similar to a website address, except instead of http:// - it will begin with ftp://.

# In this article we will discuss ABAP code that takes into account the date at which the flat file was generated. The naming convention used for the flat file is - *flatfilenameYYYYMMDD.CSV*. In our example *we will be loading files the day after they were generated and posted on the FTP server*.

### **Accessing an FTP server from within SAP BW**

SAP has fo’reseen (fɔː'siːn от foresee - предвидеть) a standard ABAP program to be able to communicate with an FTP server. This program is called ***RSFTP002*** and can be executed using transaction code SE38.

# 

# RFC destination

# RFC destination SAPFTP is used to send *a file residing locally on the user’s client directly* to a remote server without technically pulling the file through the SAP application server. However, when you want to transfer files to an SAP application server from a remote system – you need to use the SAPFTPA destination.

# In Appendix A we give the ABAP code of a program that is capable of connecting to an FTP server and transferring a file from that FTP server to the SAP BW application server. The program also includes some code to inform the user when it could not find the file it was looking for on the FTP server. This error message is useful when the program is included in a process chain. Furthermore - since we do not want our SAP BW application server to become cluttered ('klʌtə загромождать) with a myriad of flat files - a simple clean up service is foreseen.

### **Where are flat files located within BW after the transfer?**

Once you have transferred the flat files from the FTP server onto the BW application server - you can find them in the SAP home directory **DIR\_HOME**. This directory can be accessed using transaction code ***AL11***.

### **Loading flat files from the BW application server into the PSA**

Now that your flat files are available on the BW application server - you would like to load them into the PSA of BW. *Once the data is located in the PSA - you can load it further on into DSO’s | InfoCubes etc.*

First you have to define a DataSource suitable for loading the type of data in the flat file. Then make the correct selections in the fields below -

* Adapter = Load Text-Type File from Application Server
* Data Format = Separated with Separator - for example - CSV

You should also carefully define the properties of each individual field. This can be done in the Fields tab. Finally you should test your DataSource by using the ‘Read Preview Data’ button which can be found in the Preview tab.

# Once the DataSource has been created you can move on to the next step - creating an InfoPackage for carrying out the actual data load from the flat file into the PSA. It’s important to set the *Character Set Settings* to Direct Entry and to choose 1100 as the *Character Set* - 1100 is the code page used by the default installation.

# 

# Note that in the InfoPackage the path towards the flat file to be loaded is determined in a routine. This is done because the flat file name includes a date which changes every day. Assume that every night - a flat file generated the day before is loaded. The code for the path will look like

data previous\_day type sy-datum.

previous\_day = sy-datum - 1.

// 

concatenate '/usr/sap/BWP/DVEBMGS53/work/*flatfilename*' previous\_day'.csv'

into p\_filename.

p\_subrc = 0.

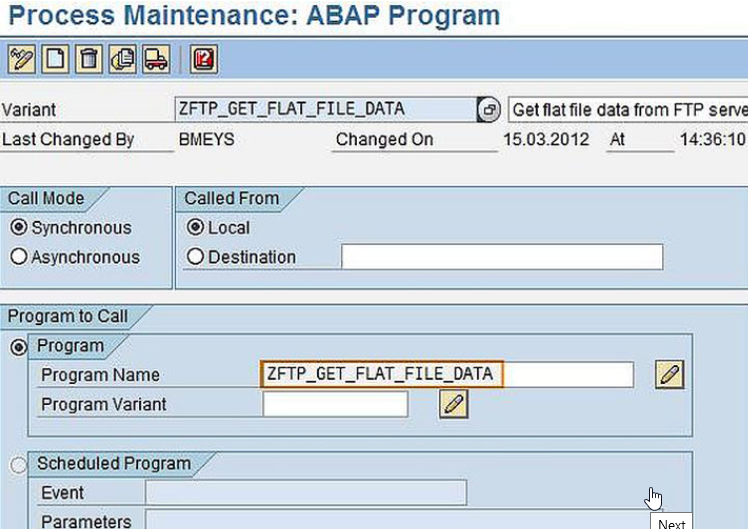
### **Bringing the pieces together in a process chain**

You can integrate the program presented in Appendix A’and the InfoPackage for flat file loading into a process chain that also contains all the steps necessary for loading the flat file data from the PSA into DSO | InfoCubes | InfoObjetcs etc.

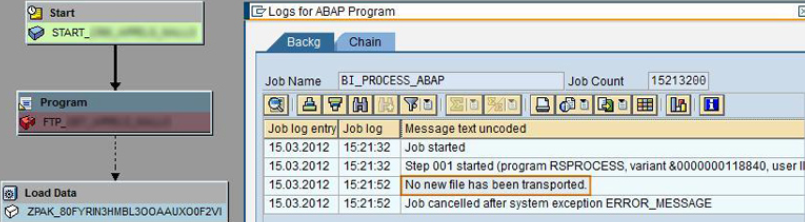
# 

The second step of the process chain includes the ABAP program for retrieving the flat file from the FTP server and copying it to SAP BW whereas step 2 represents the InfoPackage to actually load the flat file data from the SAP BW application server into the PSA. The ensuing steps can be standard processes to load data into a DSO - activating it - loading it into an InfoCube and so on.

Using transaction code RSPC - you can include a step by selecting the ABAP program process type under the *General Services* folder. All you need to do then is give the variant a proper name and select the program of Appendix A as the program to call.



When a certain step of a process chain fails - SAP usually gives you an error message describing what went wrong. Therefore we included code to generate an appropriate error message whenever the daily process chain fails due to the absence of the new flat file.



### **APPENDIX A**

report **zftp\_get\_flat\_file\_data**.

data:

list\_tab type table of abaplist,

listtxt type list\_string\_table,

file\_name(25) typec,

command\_rename(69) type c,

command\_get(29) type c,

command\_dele(30) type c,

transfer\_ok(30) typec,

ftp\_code(3) typec,

date1 type sy-datum,

date2 type sy-datum,

day type clength 2.

date1 = sy-datum - 1.

date2 = date1 - 1.

concatenate '*flatfilename*\_' date1 '.csv' into file\_name.

concatenate 'get' file\_name into command\_get separated by space.

*\* Part 1*

*\* In part 1 of the code a connection is made with the FTP server. The correct directory is selected and a* ***get***

*\* command is executed to fetch the flat file from the server. The name of the flat file has been constructed*

*\* using the system date.*

\*

submit rsftp002

with cmd1 = 'cd *directory\_name*'

with cmd2 = command\_get

with compress = 'N'

with dest = 'SAPFTPA'

with host = *'xx.xxx.xxx.xxx*'

with pwd = '*password*'

with user = '*username*'

and return

exporting list to memory.

*\* Part 2 provides some code to generate a meaningful error message in case something goes wrong.*

*\**

call function 'LIST\_FROM\_MEMORY'

tables

listobject = list\_tab

exceptions

not\_found = 1

others = 2.

if sy-subrc = 0.

call function 'LIST\_TO\_ASCI'

importing

list\_string\_ascii = listtxt

tables

listobject = list\_tab.

if sy-subrc 0.

message idsy-msgid type sy-msgty number sy-msgno

with sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.

endif.

read tablelisttxt into transfer\_ok index 8.

ftp\_code = transfer\_ok+0(3).

if ftp\_code '226'.

message e051(rsar) with'No new file has been transported.'.

endif.  
endif.

*\* Part 3 Part 3 makes sure that the flat file that was loaded during the previous run of daily process chain is*

*\* removed from the FTP directory.*

\*

clear file\_name.

concatenate '*flatfilename*\_'date2 '.csv' into file\_name.

concatenate 'dele' file\_name into command\_dele separated by space.

submit rsftp002

with cmd1 = 'cd *directory\_name*'

with cmd2 = command\_dele

with compress = 'N'

with dest = 'SAPFTPA'

with host = *'xx.xxx.xxx.xxx'*

with pwd = *'password'*

with user = *'username'*

and return

exporting list to memory.

*\* Part 4*

*\**

day = date2+6(2).

if day ne '01'.

delete datasetfile\_name.

endif.

## **Loading and Testing XML file using SPROXY transaction**

# <http://saptechnical.com/Tutorials/Others/Proxy/XML.htm>

## **Tutorial - Data exchange between ABAP and XML**

<https://www.heidoc.net/joomla/technology-science/sap-and-xslt/5-tutorial-data-exchange-between-abap-and-xml>

## **Converting an XML file with many hierarchy levels to ABAP format**

# <http://saptechnical.com/Tutorials/ABAP/XML/Index.htm>

# How to Download an iDoc to XML File (3 Different Techniques)

# <https://sapintegrationhub.com/abap/ale-idoc/how-to-download-idoc-to-xml-file/>

# XML data loading to BW

<https://answers.sap.com/questions/907250/xml-data-loading-to-bw.html>

[Содержание](#Содержание)

# − Is there a specific steps of sending XML data to BW ?

**−** Follow the steps

1. Install XML 3.0 parser.
2. Create an Info source.
3. Assign PC file as data source.
4. Create transfer and communication structure and activate.
5. From transfer structure screen select Extras --> Create BW datasource with SOAP.

After succesful generation the data source is connected to myself datamart - the name of the data source is 6A\*

1. Create an info package.
2. Load data from XML file.

# XML file data load to NW2004s-BI

<https://answers.sap.com/questions/1212718/xml-file-data-load-to-nw2004s-bi.html>

[Содержание](#Содержание)

**− I am not able to create the XML DataSource** - BW DataSource with SOAP Connection.

I tried to create the XML data source but there is no option as Extras -> Create BW Data source with SOAP Connection. Could you please help me to create the XML data source in NW2004S.

− Follow the next steps

1. Create a WebService SourceSystem.
2. Create a DataSource.
3. Generate DataSource.
4. Create InfoPackage ---> the Webservice is generated and can be used.
5. Send data via WebService to BW.
6. Data is inserted in the PSA - no Delta Queue is used anymore.

− I have the XML file at the desk top and want to load the data from that file to BI - in this case still web service data souce will work?

− A WebService data source can not pull data ⇒ the data needs to be pushed to BW via SOAP | Proxy - etc.

Take a look at – How To Push data from BI to XI including receiver examples -

<https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/e698aa90-0201-0010-7982-b498e02af76b>

# Интеграция XML данных — другой путь

<https://habr.com/ru/post/325186/>

## Creating an XML DataSource

<https://help.sap.com/doc/erp2005_ehp_06/6.0.6/en-US/0b/3bc73b24062b48e10000000a11402f/frameset.htm>

sap help load XML file to BW

sap blog load XML file to BW

**How to... Send XML Data to BW**

<https://archive.sap.com/kmuuid2/21d8aa90-0201-0010-5e83-a3798b9a5ee0/How%20to%20Send%20XML%20Data%20to%20BW>

− Кто-нибудь сталкивался с необходимостью загрузки данных в BW в формате xml.

Название полей xml постоянно – но их последовательность может меняться.

Пробовала через **web-сервис** - создала источник данных - но структура xml файла в данном случае четко определена структурой источника - мне не подходит.

Может быть с применением abap можно грузить такие файлы?

Необходимо данные из xml-файла грузить в DSO. Видимо тут только *DSO прямой записи* поможет *с преобразованием xml во внутреннюю таблицу* - ищу примеры.

Примеры xml

1.

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

<pkg form="F02" year="2014" period="2" valuta="RUB" createdate="2015-04-09T10:46:20" user="123"

rowcodetype="standard" periodtype="2" minusexpand="0">

<data>

<item rowcode="2100" valuetype="standard" value="101876"/>

<item rowcode="2110" valuetype="standard" value="5641380"/>

<item rowcode="2120" valuetype="standard" value="-5539504"/>

…

</data>

</pkg>

2.

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

<pkg adminmail="admin@admin.com" periodtype="assoi" rowcodetype="standard" user="123"

createdate="2015-03-31T13:36:25" valuta="RUB" period="16" year="2014" form="F02" >

<data>

<item value="686635" valuetype="sumsng" rowcode="2110"/>

<item value="564564" valuetype="sumsng" rowcode="2120"/>

.....

</data>

</pkg>

Нашла решение через загрузку *DSO прямой записи* с преобразованием xml во внутреннюю таблицу с использованием **call transformation**.

ABAPstatement **CALL TRANSFORMATION** - calls the specified XSL transformation -   
XSLT or a Simple Transformation - ST.

# XSLT Transformations - XML 2 ABAP - a working example with deep structures

<https://blogs.sap.com/2018/09/10/xslt-transformations-xml-2-abap-a-working-example/>

## **The XML**

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

<File>

<Head>

<Name>K. Richards</Name>

</Head>

<Items>

<ITM *QTY*="23" *EAN*="123123123123" *CAT*="BL"/>

…

<ITM QTY="50" *SOH*="4000299949" EAN="123123123123" *SOI*="000010" CAT=""/>

…

</Items>

</File>

## **The XSLT**

The import can be achieved with the below ***XSL Transformation*** - Transaction **STRANS**

<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:sap="http://www.sap.com/sapxsl" version="1.0">

<xsl:strip-space elements="\*"/>

<xsl:template match="/">

<asx:abap xmlns:asx="http://www.sap.com/abapxml" version="1.0">

<asx:values>

<FILE> <!--FILE Needs to match name of RESULT variable in CALL TRANSFORMATION call-->

<xsl:for-each select="/File">

<!-- <DUMMY1> = Dummy, not in ABAP structure-->

<DUMMY1>

<HEAD>

<NAME>

<xsl:value-of select="Head/Name"/>

</NAME>

</HEAD>

<ITEMS>

<xsl:for-each select="Items/ITM">

<DUMMY2>

<!-- <DUMMY2> = Dummy, not in ABAP structure-->

<QTY>

<!-- @ = Attribute of element rather than element-->

<xsl:value-of select="@QTY"/>

</QTY>

<SOH>

<xsl:value-of select="@SOH"/>

</SOH>

<EAN>

<xsl:value-of select="@EAN"/>

</EAN>

<SOI>

<xsl:value-of select="@SOI"/>

</SOI>

<CAT>

<xsl:value-of select="@CAT"/>

</CAT>

</DUMMY2>

</xsl:for-each>

</ITEMS>

</DUMMY1>

</xsl:for-each>

</FILE>

</asx:values>

</asx:abap>

</xsl:template>

</xsl:transform>

## **The ABAP**

And here is the ABAP Report that calls the transformation

REPORT **zjbo\_xslt\_2**.

TYPES BEGIN OF lty\_items.

TYPES qty TYPE string.

TYPES soh TYPE string.

TYPES ean TYPE string.

TYPES soi TYPE string.

TYPES cat TYPE string.

TYPES END OF lty\_items.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| qty | soh | ean | soi | cat |
|  |  |  |  |  |

TYPES BEGIN OF lty\_head.

TYPES name TYPE string.

TYPES END OF lty\_head.

TYPES BEGIN OF lty\_out.

TYPES head TYPE lty\_head.

TYPES items TYPE STANDARD TABLE OF lty\_items *WITH DEFAULT KEY*.

TYPES END OF lty\_out.

DATA lv\_xml TYPE string.

*" lt\_xml - строки xml файла*

DATA lt\_xml TYPE STANDARD TABLE OF string.

DATA ls\_out TYPE lty\_out.

*" lt\_out – заголовок + внутрення таблица из строк xml*

DATA lt\_out TYPE STANDARD TABLE OF lty\_out.

PARAMETERS p\_file TYPE string DEFAULT 'C:\link\to\test.xml'.

*"Load xml file*

cl\_gui\_frontend\_services=>gui\_upload(

EXPORTING

filename = p\_file

filetype = 'ASC'

CHANGING

data\_tab = lt\_xml

EXCEPTIONS

OTHERS = 19 ).

lv\_xml = concat\_lines\_of( lt\_xml ).

CALL TRANSFORMATION zmy\_xslt\_2

SOURCE XML lv\_xml

RESULT file = lt\_out.

cl\_demo\_output=>new( )->write\_data( lt\_out[ 1 ]-head )->write\_data( lt\_out[ 1 ]-items )->display( ).

## **Now for some pointers**

Here are some important tips

1. The RESULT variable of the CALL TRANSFORMATION **must**have the same name as the root target element in the XSLT - in this example file == <FILE>
2. The target elements in the XSLT – i.e. the elemenst that correspond with your target ABAP structure - must be in CAPITALS - like <HEAD> | <NAME> …
3. The Source element references must be in same case as in the XML - like /Items/ITM
4. The output of the CALL TRANSFORMATION must be a table type - structure is not allowed. Other outputs like XML and ref to data is also possible - see SAP Help. Like in our example - **lt\_out**is a table type.
5. Each substructure or table line in the output needs to be explicitly defined with a dummy structure name. These dummy structure names do not exist in the ABAP target but need to be defined. The name used is irrelevant. See <DUMMY1> and <DUMMY2> in XSLT.

**GUI\_UPLOAD** - is for getting the data from flat file to internal table.

**GUI\_DOWNLOAD** - is for sending the data from internal table to flat file.

# ST Transformations - XML 2 ABAP - a working example

# <https://blogs.sap.com/2019/07/02/st-transformations-xml-2-abap-a-working-example/>

# Rem - xml – тот же.

## **The ST**

Code of the ST transformation

<?sap.transform simple?>

<tt:transform xmlns:tt="http://www.sap.com/transformation-templates">

<tt:root name="FILE"/>

<tt:template>

<File tt:ref=".FILE"> <!--FILE Needs to match name of RESULT variable in CALL TRANSFORMATION call-->

<Head tt:ref="HEAD">

<Name tt:value-ref="NAME"/>

</Head>

<Items>

<tt:loop ref="ITEMS">

<ITM>

<tt:group>

<tt:cond frq="?"><tt:attribute name="QTY" value-ref="QTY"/></tt:cond>

<tt:cond frq="?"><tt:attribute name="SOH" value-ref="SOH"/></tt:cond>

<tt:cond frq="?"><tt:attribute name="EAN" value-ref="EAN"/></tt:cond>

<tt:cond frq="?"><tt:attribute name="SOI" value-ref="SOI"/></tt:cond>

<tt:cond frq="?"><tt:attribute name="CAT" value-ref="CAT"/></tt:cond>

</tt:group>

</ITM>

</tt:loop>

</Items>

</File>

</tt:template>

</tt:transform>

## **The ABAP**

Almost the same - except to output the result into a structured variable instead of an internal table - to correspond better to the input XML.

REPORT.

TYPES:

BEGIN OF lty\_items,

qty TYPE string,

soh TYPE string,

ean TYPE string,

soi TYPE string,

cat TYPE string,

END OF lty\_items.

TYPES:

BEGIN OF lty\_head,

name TYPE string,

END OF lty\_head.

TYPES:

BEGIN OF lty\_out,

head TYPE lty\_head,

items TYPE STANDARD TABLE OF lty\_items WITH DEFAULT KEY,

END OF lty\_out.

DATA:

lv\_xml TYPE string,

lt\_xml TYPE STANDARD TABLE OF string,

ls\_out TYPE lty\_out.

PARAMETERS p\_file TYPE string LOWER CASE DEFAULT 'C:\link\to\test.xml'.

"Load xml file

cl\_gui\_frontend\_services=>gui\_upload(

EXPORTING

filename = p\_file

filetype = 'ASC'

CHANGING

data\_tab = lt\_xml

EXCEPTIONS

OTHERS = 19 ).

lv\_xml = concat\_lines\_of( lt\_xml ).

CALL TRANSFORMATION zmy\_xslt\_2

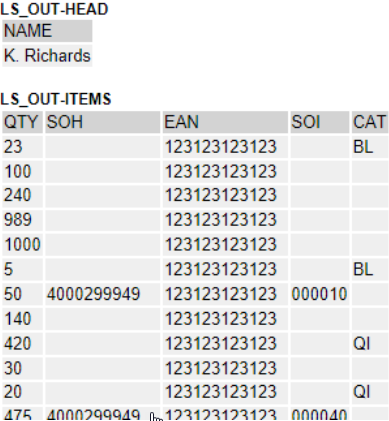
SOURCE XML lv\_xml

RESULT file = ls\_out.

cl\_demo\_output=>new( )->write\_data( ls\_out-head )->write\_data( ls\_out-items )->display( ).

## The result

running the ABAP program -> contents of variable LS\_OUT



# Файловый интерфейс – операции с файлами

Должна быть возможность обмена данными между юникодными и не юникодными системами - т.к. отображение данных в памяти отличается.

В юникодных системах требуется обязательное указание кодовой страницы перед работой через интерфейс.

− В ABAP-коде есть строки записи файла в расшаренную в сети папку и адресация к ней прописана так - [\\SERVER\Folder1\Folder2](file:///\\SERVER\Folder1\Folder2) - доступ разрешен с определенным логином и паролем.

При выполнении этого кода на клиентской машине выскакивает Windows-сообщение об ошибке *Нет доступа*. Если же в клиентской винде прописать логин/пароль к этому серваку - то все ок.

Можно ли как-то прописать путь/логин/пароль к этой папке на сервере SAP так - чтобы все клиентские обращения шли через него и у локальных пользователей никаких доп. настроек не требовалось?