ARIS PROCESS PERFORMANCE MANAGER

PPM PROCESS EXTRACTORS

VERSION 10.5.3

OCTOBER 2021

This document applies to ARIS Process Performance Manager Version 10.5.3 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Contents

1 General

This manual describes the function of the PPM SAP-2-PPM, JDBC-2-PPM and CSV-2-PPM process extractors.

From the data sets of various source system types, process extractors create XML files that can be imported into ARIS Process Performance Manager (or PPM)

Please note that this manual is not intended to replace user or customizing training. It is a source of reference containing information that supplements the information provided in the online help.

2 XML

This chapter contains basic information about XML, which is necessary to understand the subsequent chapters.

2.1 What is XML?

The abbreviation XML stands for e**X**tensible **M**arkup **L**anguage. XML is a meta-language for the description of display languages such as HTML. Meta-languages provide the rules required for the definition of document types. Display languages allow documents to be output correctly.

2.2 Structure of an XML document

An XML document is always made up of two character types: the actual data and the so-called tags or markups. Tags are XML instructions, which describe the division of the document into storage units and its logical structure. The structure itself is saved in a document type definition (DTD).

Tags are always written in pairs in angle brackets. Every start tag always has a corresponding end tag.

XML attributes are used within the tags. An attribute may only occur once within a tag.

XML documents consist of elements. An element is made up of two XML tags and the enclosed text. Blank elements consist of only one tag and always end with a slash (**/**) before the final bracket.

You can create simple XML documents with a text editor. In the following example, the DTD is specified in square brackets in the XML file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE memberlist
 [
  <!ELEMENT memberlist (no, name, age)>
  <!ELEMENT no (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT age (#PCDATA)>
 ]>
<memberlist>
  <no>001</no>
  <name>Doe, John</name>
  <age>27</age>
</memberlist>
```

If you save this document under the name of your choice with the extension **.xml**, Internet Explorer can display the document in a structured form.

2.3 PPM and XML

PPM uses XML as a universal data format. The entire configuration of the PPM system is provided by XML files.

The PPM process extractors extract data from different source systems and save it in PPM compatible XML files.

3 Overview

This chapter provides an overview of the extraction of data from application systems for use in ARIS Process Performance Manager. The extracted data can be imported into ARIS Process Performance Manager with no changes using the PPM XML import interface and processed by using suitable fragment and mapping definitions. The XML import interface is described in detail in the **PPM Data import** manual.

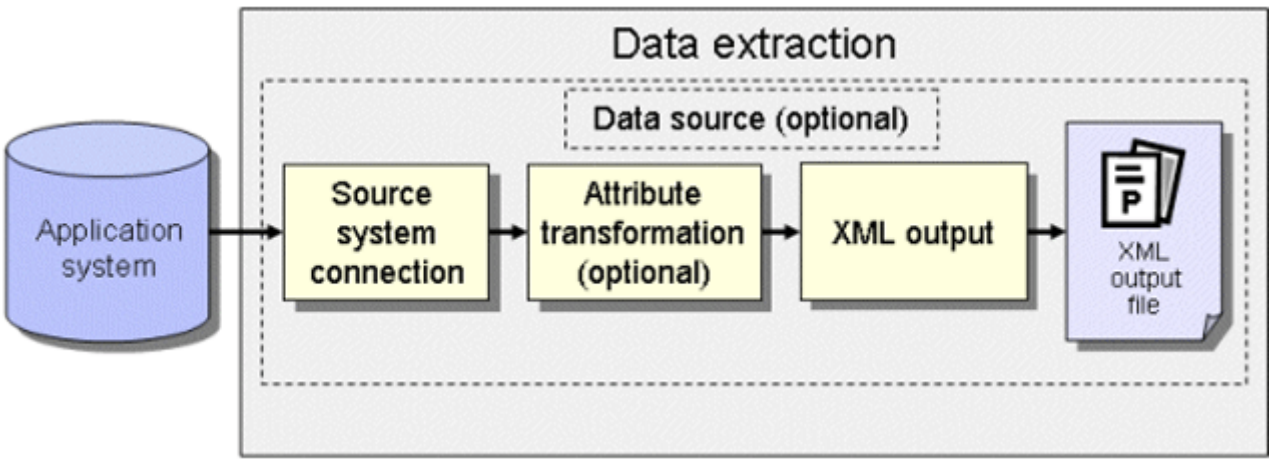
Via XML configuration files, the PPM CSV-2-PPM, SAP-2-PPM and JDBC-2-PPM process extractors are adapted to the source system from which the data is to be extracted.

3.1 Data extraction

This chapter provides an overview of the extraction of data from application systems for use in ARIS Process Performance Manager. The extracted data can be imported into ARIS Process Performance Manager with no changes using the PPM XML import interface and processed by using suitable fragment and mapping definitions. The XML import interface is described in detail in the **PPM Data import** manual.

Via XML configuration files, the PPM CSV-2-PPM, JDBC-2-PPM and SAP-2-PPM process extractors are adapted to the source system from which the data is to be extracted.

The following general representation illustrates the basic functioning of data extraction from source systems, the subsequent data transformation and the output in XML output files.



SOURCE SYSTEM CONNECTION

The source system connection is created by specifying a system configuration containing the required access data for the source system to be extracted, for example, system number, access type, and access password of the source system user.

You need to ensure that the specified source system user has appropriate access authorization to extract the relevant data fields.

ATTRIBUTE TRANSFORMATION (OPTIONAL)

If required, source system attributes can be modified before being imported into the PPM system and attribute types can be added and transformed. An appropriate XML configuration needs to be created for attribute transformation.

XML OUTPUT

The extracted data is written to output files in a PPM-compatible XML format.

DATA SOURCE (OPTIONAL)

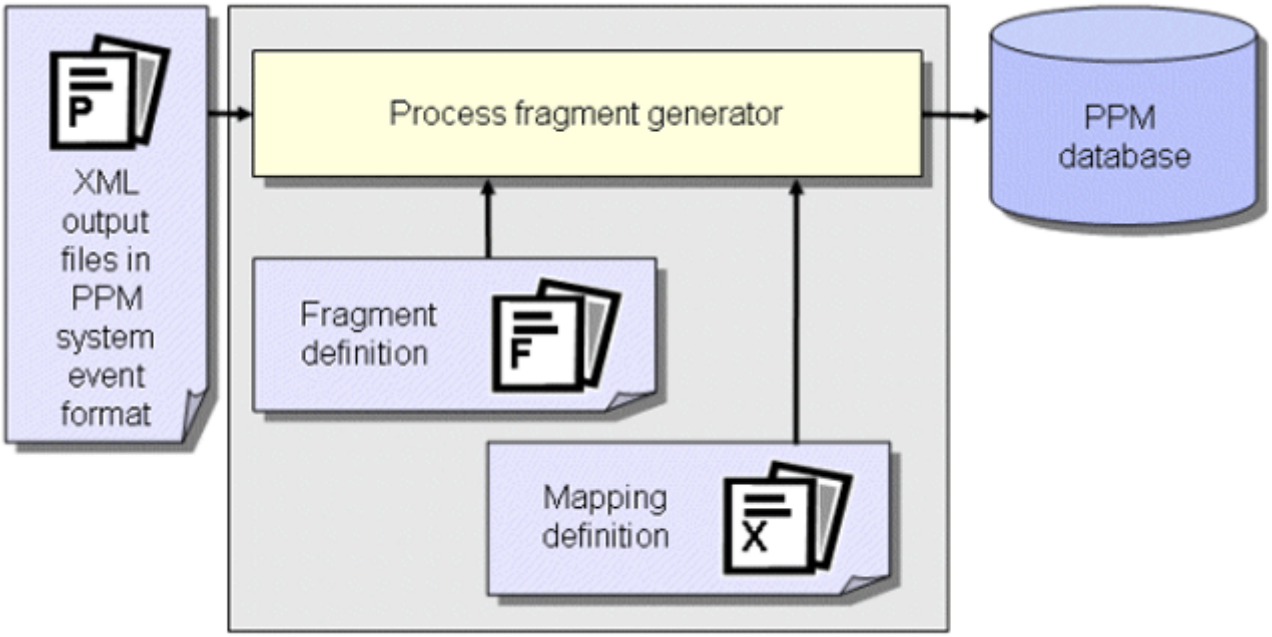
All XML files required for the extraction as well as the output file(s) can be specified in an XML file.

3.2 Data import

In addition to the XML output file in PPM system event format, a file in process instance-independent measure format or in dimension data format can also be output and imported into a PPM system. Only the import of the XML output files in PPM system event format is dealt with here.

Each system event in the XML output file generated is assigned a fragment definition when imported and is instantiated in the PPM database. The source system attributes specified in the mapping are copied to the objects in this fragment instance. The fragment instances are then saved in the PPM database.

Schema: Data import in PPM system event format



In a further work step, the fragment instances imported into the database are compiled into process instances using the **runppmimport** PPM command and assigned to process types. After measure calculation, the process instances are available for detailed analyses.

Tip

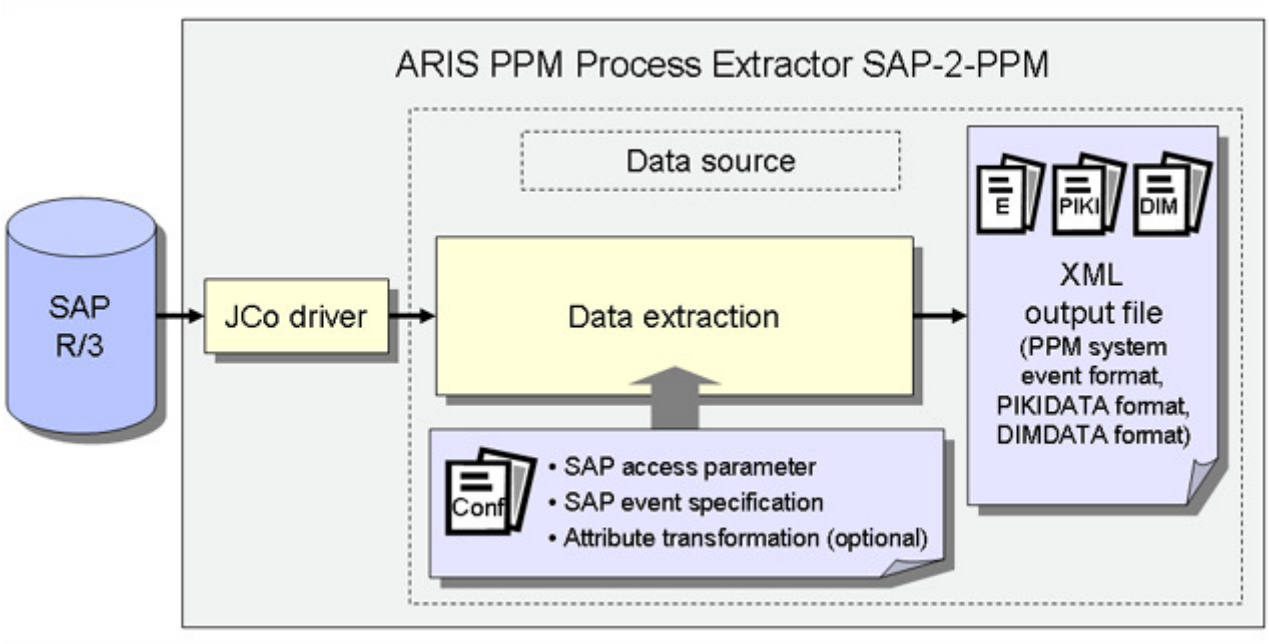
- For further information about the configuration of the XML import please refer to the **PPM Data Import** manual.
- The configuration of all files relevant for **runppmimport** is described in detail in the **PPM Customizing** manual.

4 PPM Process Extractor SAP-2-PPM

This chapter provides an overview of the architecture, functioning and configuration of PPM Process Extractor SAP-2-PPM.

4.1 Architecture

The figure below illustrates the functionality of PPM Process Extractor SAP-2-PPM:



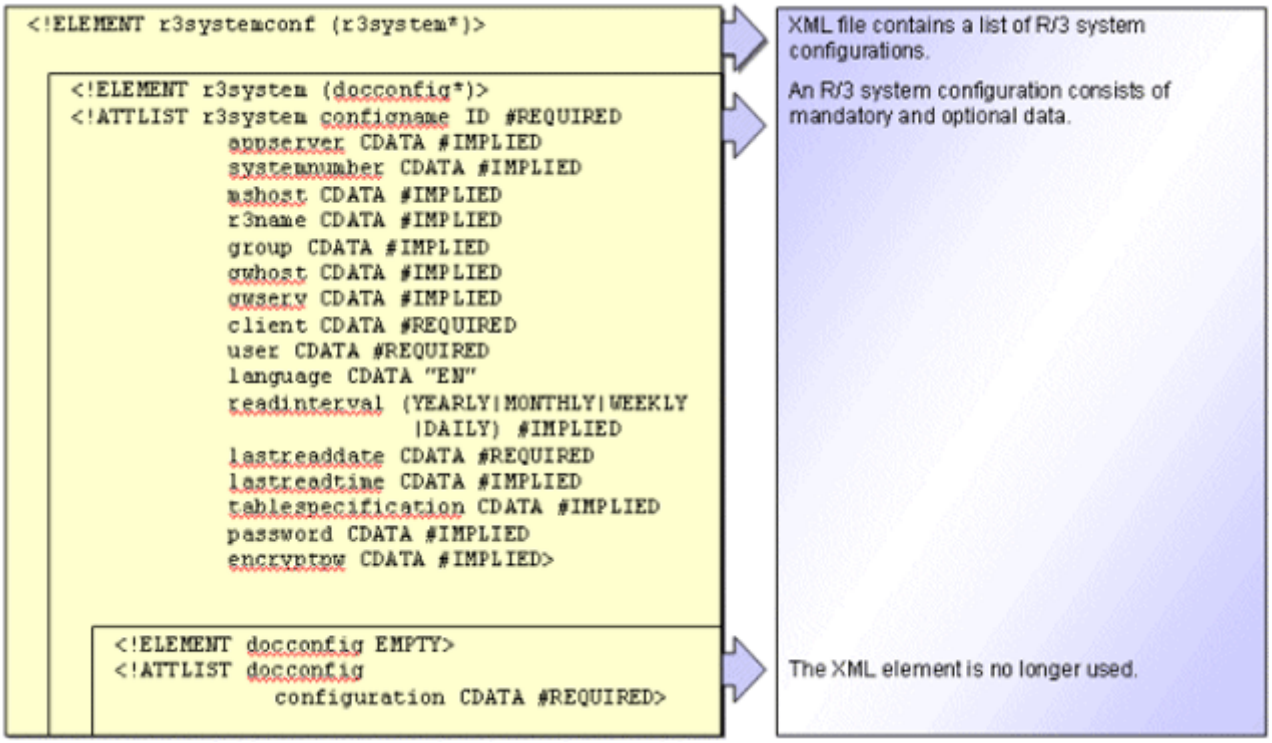
The Java Connector driver (JCo) is used to create a connection to the SAP R/3 system with the access data for the R/3 system configuration. Data is then extracted from the tables in the R/3 system with the R/3 table configuration and an optional attribute transformation using the **data extraction** (page 4) module and is written to XML files in a PPM-compatible output format.

You need to ensure that the specified source system user has appropriate access authorization to extract the relevant data fields and that the requirements for the R/3 source system are met. If you have any doubts or concerns, please contact your SAP administrator.

4.2 R/3 system configuration

The access data for the R/3 system is specified in an XML file. The name of this XML file is transferred to the command line tool as an argument.

The format of the XML file is specified by the following DTD:



XML attribute name	Description	Example
configname	Unique name of the configuration	ides_doc_vdap
appserver	Computer name or IP address of source system computer	172.20.210.21
systemnumber	SAP system number	00
mshost	Name of the SAP message host	/H/154.37.28.6/S/4435
r3name	R/3 system name	A20
group	Name of application server group	A20_ALL
gwhost	Computer name of R/3 gateway	172.20.210.32
gwserv	Service number of the R/3 gateway	6789
client	Name of the SAP client	100

XML attribute name	Description	Example
user	Name of the SAP system user	sapuser
language	Language of text fields extracted	DE
readinterval	No longer used. Still included for compatibility reasons.	YEARLY
lastreaddate	Date from which data is extracted. Not applicable when using a **data source** (page 158), format: **yyyymmdd**	19700101
lastreadtime	Time from which data is extracted. Not applicable when using a **data source** (page 158), format: **hhmmss**	125708
tablespecification	No longer used. Still included for compatibility reasons.	
password	Access password for R/3 system	mypassword
encryptpw	Access password for R/3 system in encrypted form	

The first time you enter an R/3 system into the system configuration or change the password, specify the password in the **password** XML attribute. The first time the R/3 system is accessed (for example, connection test, starting extraction), the password is encrypted and saved in the **encryptpw** XML attribute. The value of the **password** XML attribute is deleted.

The type of access to an R/3 system is determined by specifying the value of XML attributes.

R/3 system type	XML attributes	Example
Application server	appserver systemnumber	... configname="..." appserver="192.168.102.100" systemnumber="01" mshost="" r3name="" group="" gwhost="" gwserv="" ...
Message Host	mshost r3name group	... configname="test_group_sd" appserver="..." systemnumber="03" mshost="/H/156.243.123.6/H/154.34.37.2/S/4435/H/saproutertest/H/112.4.3.2" r3name="A20" group="A20_ALL" gwhost="" gwserv="" ...
Gateway server	appserver systemnumber gwhost gwserv	... configname="..." appserver="172.20.212.232" systemnumber="01" mshost="" r3name="" group="" gwhost="172.20.212.240" gwserv="6789" ...

Example

```
...
<r3system configname="ides_doc_vbap" appserver="172.20.210.211"
          systemnumber="00" client="100" user="sapuser"
          language="DE" readinterval="YEARLY"
          lastreaddate="19700101" password="testpassword">
     <docconfig configuration="SD_VBAP"/>
     <docconfig configuration="SD_MSEG"/>
</r3system>
...
```

4.3 R3 table configuration (system event specification)

The R/3 table configuration specifies which table fields are extracted from the R/3 system and written to the system events as source system attributes. You can save several table configurations with unique names in the XML file.

The R/3 table configuration consists of the following components:

GLOBAL TABLES

Global tables are used to extract information that is written for all system events.

FOREIGN KEY TABLES

The **docreftable** XML element contains the name of the foreign key table. It specifies how the data area to be extracted from the system event table is limited. The primary key fields specified in the **pkfield** XML element link the foreign key table to the system event table and other foreign key tables.

SYSTEM EVENT TABLE

The **doctable** XML element contains the name of the system event table. It specifies the documents to be extracted for a document flow. Each data record extracted from the system event table generates a system event in the output file (**event** XML element).

DATA TABLES

The information in the system event table can be supplemented by extracting additional data fields from any other data tables (for example, the material number is extracted from the system event table and the descriptive text relating to this number is extracted from a data table).

The following XML file framework illustrates the configuration of the tables from which data is to be extracted. The chapter on **table access configuration** (page 100) describes which XML elements and attributes are optional.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE r3systemconffields SYSTEM 'xmlextractor_tableconfiguration.dtd'>

<xmlextractor_tableconfiguration>
  <configuration name="..." printname="..." classtouse="...">
    <globaltable name="..." tablename="..." classtouse="...">
      <fieldtoread name="...">
        <textref tablename="..." reffieldname="..."
                 textfieldname="..." langfieldname="..."/>
      </fieldtoread>
      ...
    </globaltable>
    <docspec>
      <docreftable name="..." tablename="..."
                                  classtouse="...">
```

```
    <condition fieldname="..."
                          logicaloperator="..." >
      <value>...</value>
    </condition>
    <pkfield name="..." fktablename="..."
            fkfieldname="..." logicaloperator="...">
      <fkpart readfrom="..." startposition="..."
                                  length="..."/>
      <prefix>
        <value>...</value>
      </prefix>
      <postfix>
        <value>...</value>
      </postfix>
    </pkfield>
    ...
  </docreftable>
  ...
  <doctable name="..." tablename="..."
                classtouse="..." >
    <condition fieldname="..." logicaloperator="..." >
      <value>...</value>
    </condition>
    <pkfield name="..." fktablename="..."
           fkfieldname="..." logicaloperator="...">
      <fkpart readfrom="..." startposition="..."
                             length="..."/>
      <prefix>
        <value>...</value>
      </prefix>
      <postfix>
        <value>...</value>
      </postfix>
    </pkfield>
    ...
    <fieldtoread name="...">
      <textref tablename="..." reffieldname="..."
           textfieldname="..." langfieldname="..."/>
    </fieldtoread>
    ...
  </doctable>
</docspec>
<table name="..." tablename="..."
                               classtouse="...">
  <condition fieldname="..." logicaloperator="...">
     <value>...</value>
  </condition>
  <pkfield name="..." fktablename="..."
            fkfieldname="..." logicaloperator="...">
     <fkpart readfrom="..." startposition="..."
                                  length="..."/>
     <prefix>
       <value>...</value>
     </prefix>
     <postfix>
       <value>...</value>
```

```
        </postfix>
      </pkfield>
      ...
      <fieldtoread name="...">
         <textref tablename="..." reffieldname="..."
           textfieldname="..." langfieldname="..."/>
       </fieldtoread>
      ...
    </table>
    ...
  </configuration>
  ...
</xmlextractor_tableconfiguration>
```

4.3.1 Global meta data

The **globaltable** XML element can be used to transfer meta data from the PPM SAP-2-PPM and JDBC-2-PPM process extractors to the output file as global system event attributes. Typical data for global system event attributes include the names of the configurations that were used to extract the data. The data itself is determined from the argument values specified in the command line. For arguments that are not specified in the command line, the default value is determined.

The identification of keywords is case-sensitive.

Global system event attributes apply to all system events in the current output file.

Keyword	Value from command line argument
SYSTEM_CONFIG_FILE_NAME	-systemconfig <filename> ...
SYSTEM_CONFIG_NAME	-systemconfig ... <configname>
TABLE_CONFIG_FILE_NAME	-tableconfig <filename> ...
TABLE_CONFIG_NAME	-tableconfig ... <configname>
BEGIN_DATE,	-begindate <dd.MM.yyyy>
BEGIN_TIME	-begintime <hh:mm:ss>
END_DATE	-enddate <dd.MM.yyyy>
END_TIME	-endtime <hh:mm:ss>
CPD	-cpd <int>
OUT_FILE_NAME	-outfile <file name>
PIKI_DATA_MAPPING_FILE_NAME	-pikidatamapping <filename>...

Keyword	Value from command line argument
PIKI_DATA_MAPPING_NAME	-pikidatamapping ... <pcname>
DIM_DATA_MAPPING_FILE_NAME	-dimdatamapping <filename> ...
DIM_DATA_MAPPING_NAME	-dimdatamapping ... <dimname>

If no value can be determined for a keyword, the keyword itself will be written to the output file. In this way you can transfer constants to the output file.

Example

The file extract below shows the configuration that is used to create global system event attributes.

```
<globaltable name="INFO">
  <fieldtoread name="SYSTEM_CONFIG_NAME"/>
</globaltable>
```

The configuration name specified in the command line using -systemconfig **<configname>**, for example, **sapppm** is written to the output file as the **INFO-SYSTEM_CONFIG_NAME** global system event attribute:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <attribute type="INFO-SYSTEM_CONFIG_NAME ">
    sapppm
  </attribute>
  ...
  <event>
    ...
  </event>
  ...
</eventlist>
```

By specifying a suitable Java class in the **classtouse** XML attribute, data can also be extracted from any other source system table.

4.3.2 Sort system events

You can sort the system events in the XML output in ascending alphanumeric order based on the content of fields you define as sorting criteria. It is possible to specify multiple fields as sorting criteria. When sorting, the fields are prioritized from the first specified to the last specified, that is, the data is first sorted based on the content of the first field specified, then by the content of the second field specified and so on.

Tip

You can use the sorting option to create a more efficient XML import of the extracted data into the PPM system. The fact that system events belonging to a process instance follow each other immediately due to sorting in the XML output files means that they can be imported directly into an EPC using the process key during the XML import.

In the table configuration, in the **parameter** XML element, fields are specified as sorting criteria for the system event table (**doctable**) and the referenced foreign key tables (**docreftable**).

Example (extract from table configuration)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="BANF">
    <docspec>
      <doctable name="EBAN">
        <parameter name="ORDER_BY">
          <value>BNFPO</value>
          <value>BANFN</value>
        </parameter>
        <condition fieldname="ERDAT"
                   logicaloperator="creationtimestamp">
          <value>yyyyMMdd</value>
        </condition>
        <pkfield name="BANFN"/>
        <pkfield name="BNFPO"/>
        <pkfield name="ERDAT"/>
        <fieldtoread name="BSART"/>
        <fieldtoread name="KONNR"/>
        <fieldtoread name="KTPNR"/>
        <fieldtoread name="LIFNR"/>
        <fieldtoread name="MFRNR"/>
      </doctable>
    </docspec>
  </configuration>
</xmlextractor_tableconfiguration>
```

As specified in the **BANF** configuration (**parameter** XML element), the system events are written to the output file(s) sorted in ascending order by the item number in the purchase requisition number (**BNFPO** field) and, as a second priority, by the purchase requisition number (**BANFN** field).

The table below shows all of the configuration options:

XML element/attribute	Description
parameter	Specifies a sorting parameter for a system event table (**doctable**) or a foreign key table (**docreftable**).

XML element/attribute	Description
name	Parameter name. It is essential to specify **ORDER_BY**.
value	Specifies a sorting field. The field must be specified as a primary key field (**pkfield**).

You can set the sorting parameter in PPM Customizing Toolkit. To do this, activate the **System event** tab in the **Data extraction** module of the **Process merge** module group. Switch to editing mode and select **Edit system event** from the pop-up menu of the system event table. In the **Specify parameter** step, you can specify your settings.

4.3.3 Add up values of a field

Depending on the data type, you can add up the values of a field either as integers or decimals by specifying parameters in the relevant **table** element in the table configuration. A separate class must be used to do this.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SUM">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP"
       classtouse="com.idsscheer.ppm.xmlextractortools.
            extractor.sap2ppm.ZNumberOperation_sap2ppm">
      <parameter name="FIELDS">
        <value>BMENG</value>
        <value>WMENG</value>
      </parameter>
      <parameter name="OPERATION">
        <value>SUM_DECIMAL</value>
      </parameter>
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
```

```
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

The example configuration specifies that for each of the floating point number fields **BMENG** and **WMENG**, the total of several values belonging to a sales document item (**POSNR**) in a sales document (**VBELN**) is calculated and written to a new system event attribute. The name of the new attribute is made up of **<tablename>-<operator><fieldname>**, for example, **VBEP-SUM_DECIMALWMENG**. The **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZNumberOperation_sap2ppm** class is used for adding up field values.

The following item numbers exist in the SAP system for the sales document **5696**:

VBELN	POSNR	WMENG	BMENG
5696	10	353,000	103,000
5696	10	215,000	52,000
5696	10	127,000	313,000
5696	20	89,000	79,000
5696	20	456,000	29,000
5696	30	23,000	269,000
5696	30	87,000	159,000
5696	30	124,000	256,000
5696	30	59,000	237,000

When extracting using the example configuration, for each item number the decimal sums are calculated for the **BMENG** and **WMENG** fields and the following system events are generated:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAP-POSNR">000010</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">468.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">695.000</attribute>
</event>
<event>
  <attribute type="VBAP-POSNR">000020</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">108.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">545.000</attribute>
</event>
<event>
```

```
  <attribute type="VBAP-POSNR">000030</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">921.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">293.000</attribute>
</event>
</eventlist>
```

4.3.4 Extract multi-valued fields

Appropriate configuration of the relevant **table** element in the table configuration enables all different values of a field to be extracted (**fieldtoread**) to be written to a system event. A separate class must be used to do this.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="MULTIPLE_VALUES">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.sap2ppm.
                       ZTableMultipleValues_sap2ppm">
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
      <fieldtoread name="WMENG"/>
      <fieldtoread name="EDATU"/>
    </table>
    <table name="VBAK" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.sap2ppm.
                       ZTableMultipleValues_sap2ppm">
      <pkfield name="VBELN" fktablename="VBEP"
                              fkfieldname="VBELE"/>
      <fieldtoread name="VBTYP"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

In the example configuration, using the **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableMultipleValues_sap2ppm** class for all data tables (**table** elements) specifies that all fields to be extracted (**fieldtoread** elements) are extracted with multiple values if multiple

values exist for a field in the source system database. In the XML output file, the extracted field with multiple values appears as an attribute with the same name in a system event:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004969</attribute>
    <attribute type="VBEP-EDATU">19970103</attribute>
    <attribute type="VBEP-EDATU">19970107</attribute>
    <attribute type="VBEP-WMENG">138.000</attribute>
    <attribute type="VBEP-WMENG">317.000</attribute>
    <attribute type="VBEP-WMENG">496.000</attribute>
  </event>
</eventlist>
```

4.3.5 Concatenate new attribute values from extracted attribute values

The **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableConcatFKs_sap2ppm** class can be used when extracting data from an SAP system to generate new system event attributes by concatenating the values of attributes already extracted. The new system events created can be referenced in other data table configurations (**table** XML element).

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.ppm.
                        xmlextractortools.extractor.
                    sap2ppm.ZTableConcatFKs_sap2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
```

```
  </configuration>
</xmlextractor_tableconfiguration>
```

If you are using the **ZTableConcatFKs_sap2ppm** class, you must specify exactly one **fieldtoread** element. The element contains the name of the system event attribute to be created. In the example, a new system event attribute with the name **RBKP_NEW-AWKEY_GENERATED** is created. The value of the attribute consists of the values of the **pkfield** elements specified.
The normal **fkpart**, **prefix** and **postfix** operations can be used for the **pkfield** elements.

A system event created with the above configuration could look like this:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED">51056012042005
  </attribute>
</event>
...
```

Special case

In the **table** configuration, the special **pkfield** element **CLIENT** can be used to automatically add the number of the SAP client when extracting data. The **name**, **fktablename** and **fkfieldname** XML attributes must have the value **CLIENT**.

Example (including postfix element)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEWER" classtouse="com.idsscheer.
                  ppm.xmlextractortools.extractor.
                   sap2ppm.ZTableConcatFKs_sap2ppm">
      <pkfield name="CLIENT" fktablename="CLIENT"
                             fkfieldname="CLIENT">
        <postfix>
          <value> test </value>
        </postfix>
      </pkfield>
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
```

```
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

A system event that is created when extracting data from client 800 could look like this:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEWER-AWKEY_GENERATED"
              >800 test 51056012042005</attribute>
</event>
...
```

The new attribute – **RBKP_NEWER-AWKEY_GENERATED** – can then be referenced in other **table** configurations in the same way as every other SAP table field.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.
                ppm.xmlextractortools.extractor.
                   sap2ppm.ZTableConcatFKs_sap2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
    <table name="BKPF">
      <pkfield name="AWKEY" fktablename="RBKP_NEW"
                    fkfieldname="AWKEY_GENERATED"/>
      <fieldtoread name="BELNR"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

A system event created with this configuration could look like this:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED"
                      >5105601204 2005</attribute>
  <attribute type="BKPF-BELNR">5100004691</attribute>
</event>
...
```

4.3.6 Link tables

The header, document, and data tables are linked to one another by primary key relations, which are specified in the **pkfield** XML element.

XML element	XML attribute	Description
pkfield	name	Name of the primary key (table column name)
	fktablename	Identifier of the table referenced by a foreign key relation
	fkfieldname	Name of the foreign key (table column name)
	logicaloperator	Valid values: **eq**, **neq**, **gt**, **geq**, **lt**, **leq**. Default value: **eq**
prefix		String inserted before the extracted foreign key value
postfix		String inserted after the extracted foreign key value
fkpart	readfrom (optional)	Direction of the substring forming the foreign key Valid values: **left**, **right** Default value: **left**
	startposition	Position from which the substring is formed
	length (optional)	Length of substring Default value: Start or end of string, depending on the value of the **readfrom** XML attribute

XML element	XML attribute	Description
value		Value specified for conditions, prefixes, and postfixes

Example

The file extract below shows the configuration that is used to extract the **VBELN** and **POSNR** database fields from the **VBAP** table. The foreign key relations are established by the content of the **TABKEY** field in the **CDPOS** table.
The first ten characters in the **TABKEY** field contain the value that is assigned to the **VBELN** field in the **VBAP** table and the following six characters contain the value that is assigned to the **POSNR** field in the **VBAP** table.

```
<table name="VBAP">
   <pkfield name="VBELN" fktablename="CDPOS"
                         fkfieldname="TABKEY">
      <fkpart startposition="0" length="10"/>
   </pkfield>
   <pkfield name="POSNR" fktablename="CDPOS"
                         fkfieldname="TABKEY">
      <fkpart startposition="10" length="6"/>
   </pkfield>
   <fieldtoread name="VBELN"/>
   <fieldtoread name="POSNR"/>
</table>
```

4.3.6.1 Comparison of foreign key relations

By default, the foreign key relations specified in the **pkfield** XML element are checked for equality (default value: **logicaloperator="eq"**). This applies to the corresponding definitions in the document table (**doctable**), the document header table (**docreftable**) and the data table (**table**).
You can perform other comparisons by specifying the following operators:

- **neq** (not equal to)
- **lt** (less than)
- **gt** (greater than)
- **geq** (greater than or equal to)
- **leq** (less than or equal to)

The comparisons are performed lexicographically in line with the Unicode character list, that is, the value **00999** is less than the value **9**. Alphanumeric text values can also be compared.

Example

The associated reference period for each **EEINV-EINZDAT** date value is to be determined. This value represents a time interval with a start and end date. All three date fields (**EEINV-EINZDAT**, **EANLH-VON**, **EANLH-BIS**) are to be extracted. The file extract below shows the corresponding configuration with the foreign key comparisons **less than or equal to** or **greater than or equal to** for extracting field values from the **EANLH** table:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
               'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="ISU">
    <docspec>
      <doctable name="EEINV">
        <pkfield name="EINZBELEG"/>
        <pkfield name="VERTRAG"/>
      </doctable>
    </docspec>
    <table name="EEINV">
      <pkfield name="EINZBELEG" fktablename="EEINV"
                    fkfieldname="EINZBELEG"/>
      <pkfield name="VERTRAG" fktablename="EEINV"
                    fkfieldname="VERTRAG"/>
      <fieldtoread name="ANLAGE"/>
      <fieldtoread name="EINZDAT"/>
    </table>
    <table name="EANLH">
      <pkfield name="ANLAGE" fktablename="EEINV"
                    fkfieldname="ANLAGE"/>
      <pkfield name="VON" fktablename="EEINV"
           fkfieldname="EINZDAT" logicaloperator="leq"/>
      <pkfield name="BIS" fktablename="EEINV"
           fkfieldname="EINZDAT" logicaloperator="geq"/>
      <fieldtoread name="VON"/>
      <fieldtoread name="BIS"/>
      <fieldtoread name="TARIFTYP"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

A system event extracted with this configuration could look like this:

```
<event>
  <attribute type="EEINV-EINZBELEG">000000800002</attribute>
  <attribute type="EEINV-VERTRAG">00012532</attribute>
  <attribute type="EEINV-ANLAGE">00045678</attribute>
  <attribute type="EEINV-EINZDAT">20041015</attribute>
  <attribute type="EANLH-VON">20041001</attribute>
  <attribute type="EANLH-BIS">20050930</attribute>
  <attribute type="EANLH-TARIFTYP">YELLOW-PRIVAT</attribute>
</event>
```

4.3.7 Extraction using conditions

Extracting data from the tables can be limited by the use of conditions. For example, you can use conditions to ensure that when extracting data from a table, only rows corresponding to a particular date field will be extracted. The conditions are specified directly in the configuration of the table specification in the **booleancondition** or **condition** XML element and only apply to the associated table.

A condition contains the name of the data field, the comparison operator, and a concrete value. Conditions can be linked with any degree of complexity (**booleancondition** XML element).

XML element	XML attribute	Description
booleancondition (optional)	logicaloperator	Logical operators: **AND**, **OR**, **NOT** Default value: **AND**
condition (optional)	logicaloperator	Comparison operators: **eq**, **neq**, **in**, **notin**, **num_gt**, **num_geq**, **num_lt**, **num_leq**, **like** Operators for restricting the data range to be extracted: **creationtimestamp**, **valueconstraint** Default value: **eq**
	fieldname	Table field name

The **docbooleancondition** and **doccondition** XML elements enable conditional extraction from a data table depending on table fields already extracted. They are configured in a similar way to **booleancondition** and **condition**. The table name and the name of the column containing the data fields already extracted are specified in the **tablename** and **fieldname** XML attributes.

XML element	XML attribute	Description
docbooleancondition (optional)	logicaloperator	Logical operator: **AND**, **OR**, **NOT** Default value: **AND**
Doccondition (optional)	logicaloperator	Comparison operator: **eq**, **neq**, **in**, **notin**, **exists**, **notexists** Default value: **eq**

XML element	XML attribute	Description
	tablename	Name of the table containing the table fields already extracted
	fieldname	Column name

4.3.7.1 Condition operators

Common operators

The following comparison operators are supported by the **condition** and **doccondition** XML elements:

Operator	Description
eq	The field content is equal to the specified value. (Case-sensitive Java string comparison)
neq	The field content is not equal to the specified value. (Case-sensitive Java string comparison)
in	Field content is equal to a specified value from a set of values. (Case-sensitive Java string comparison)
notin	Field content is not equal to a specified value from a set of values. (Case-sensitive Java string comparison)

Condition operators

Operator	Description
num_gt	Field content is greater than specified value.
num_geq	Field content is greater than or equal to the specified value.
num_lt	Field content is less than specified value.

Operator	Description
num_leq	Field content is less than or equal to the specified value.
like (for alphanumeric R/3 data types only, that is, ACCP, CHAR, CLNT, CUKY, LCHR, NUMC, UNIT, VARC, TIMS, or DATS dictionary types)	Comparison of field values with a variable string The following placeholders are permitted: ***** No or any number of characters **?** Any single character **** Masking character for searching for placeholders or masking characters in the form: **** or ***** or **\?** Example: `<condition fieldname="OBJECTID" logicaloperator="like"> <value>*10?0\\20?0*</value> </condition>` The search is performed for values such as 5551050\20106667 or 1080\204044 but not 34510550\2030*
creationtimestamp	Time stamps (date and time) are extracted from R/3 fields with times. Their values form the basis for restricting the data range to be extracted with the **-begindate** (**-begintime**) or **-enddate** (**-endtime**) command line parameters [see chapter **Source system-specific arguments** (page 64)]. Multiple fields are separated by the character combination **#-#**. For each field, the format descriptions of the source system fields are specified in the **value** XML elements. Example (for **VBAP** table): `<condition fieldname="ERDAT#-#ERZET" logicaloperator="creationtimestamp"> <value>dd.MM.yyyy</value> <value>HH:mm:ss</value> </condition>`

Operator	Description
valueconstraint	Specifies an R/3 field with integer values that are used to delimit the data range to be extracted with the **-valueconstraint** command line parameter [see chapter **Source system-specific arguments** (page 64)]. The field to be extracted must be of the NUMC or INT4 data type. Example (for **VBAP** table): `<condition fieldname="POSNR" logicaloperator="valueconstraint" />`

doccondition operators

Operator	Description
exists	Field exists.
notexists	Field does not exist.

4.3.7.2 Complex conditions

The file extract below illustrates nested conditions using the example of the document header table:

```
...
  <docreftable tablename="VBAK" >
    <booleancondition logicaloperator="AND">
      <condition fieldname="ERDAT#-#ERZET"
                 logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="VBTYP" logicaloperator="in">
        <value>C</value>
        <value>K</value>
      </condition>
      <booleancondition logicaloperator="OR">
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>1000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="eq">
```

```
          <value>0041</value>
        </condition>
      </booleancondition >
      <booleancondition logicaloperator="AND">
        <condition fieldname="VKORG" logicaloperator="eq">
          <value>2000</value>
        </condition>
        <condition fieldname="VKBUR" logicaloperator="neq">
          <value>0060</value>
        </condition>
      </booleancondition >
    </booleancondition >
  </booleancondition >
  ...
</docreftable>
...
```

In terms of propositional logic, the file extract shown corresponds to the following condition:

ERDAT and ERZET contain the creation time
and
VBTYP is C or V
and
((VKORG is equal to 1000 **and** VKBUR is equal to 0041) **or** (VKORG is equal to 2000 **and** VKBUR is not equal to 0060)).

4.3.8 Extract database fields

The database fields in the document table and the linked data tables, from which values are to be extracted, are specified in the **fieldtoread** XML element. For each **fieldtoread** XML element, a line of the form

```
<attribute type="Type">Value</attribute>
```

is written to the XML output file.

Type is made up of the table name, the data field name, and an optional text field name. **Value** is the value extracted from the corresponding data field. All values are written in text form.

Optionally, instead of the direct data field value, the value of the referenced table (**textref** XML element) can be extracted. The optional **langfieldname** specification extracts the language-specific text of the data field.

Supplementary information is extracted from the data tables using primary key relations.

XML element	XML attribute	Description
fieldtoread	name	Name of the table column containing the data field to be extracted
textref	tablename	Table name of the referenced data table
	reffieldname	Name of foreign key
	textfieldname	Name of the table column containing the data field to be extracted
	langfieldname (optional)	Name of the table column containing the language-specific data field to be extracted
fkpart	readfrom (optional)	Direction of the substring forming the foreign key Valid values: **left**, **right** Default value: **left**
	startposition	Position from which the substring is formed
	length (optional)	Length of substring Default value: Start or end of the string (**readfrom** XML attribute)

Before data is actually extracted, a check is run as to whether the configured tables and data fields exist and whether the specified system user has appropriate access authorization.

Example

The extracted field values are written to the XML output file as attributes of a system event.

Unlocalized table fields:

```
...
  <attribute type="VBAP-WERKS">SB</attribute>
...
```

Localized table fields:

```
...
  <attribute type="VBAP-WERKS-NAME">Saarbrücken</attribute>
...
```

4.3.9 Example: XML configuration and output file

This chapter uses a practical example to illustrate the table configuration for extracting data from an SAP system and shows an extract from the XML output file generated.

4.3.9.1 Table configuration

The file extract below shows the **SD_C** table configuration for extracting data from an SAP SD system:

```
...
<xmlextractor_tableconfiguration>
  <configuration name="SD_C">
    <docspec>
      <docreftable name="VBAK">
        <booleancondition>
          <condition fieldname="ERDAT#-#ERZET"
               logicaloperator="creationtimestamp">
            <value>yyyyMMdd</value>
            <value>HHmmss</value>
          </condition>
          <condition fieldname="VBTYP"
                           logicaloperator="eq">
            <value>C</value>
          </condition>
        </booleancondition>
        <pkfield name="VBELN" />
      </docreftable>
      <doctable name="Orders" tablename="VBAP">
        <pkfield name="VBELN" fktablename="VBAK"
                           fkfieldname="VBELN"/>
        <pkfield name="POSNR" />
        <fieldtoread name="ERDAT"/>
        <fieldtoread name="ERZET"/>
        <fieldtoread name="MATNR">
          <textref tablename="MAKT" reffieldname="MATNR"
            textfieldname="MAKTX" langfieldname="SPRAS"/>
        </fieldtoread>
        <fieldtoread name="KONDM">
          <textref tablename="T178T" reffieldname="KONDM"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
        </fieldtoread>
        <fieldtoread name="SPART">
          <textref tablename="TSPAT" reffieldname="SPART"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
        </fieldtoread>
        <fieldtoread name="WERKS">
          <textref tablename="T001W" reffieldname="WERKS"
                           textfieldname="NAME1"/>
        </fieldtoread>
        <fieldtoread name="CHARG"/>
        <fieldtoread name="PSTYV"/>
```

```
      <fieldtoread name="ERNAM"/>
      <fieldtoread name="NETWR"/>
    </doctable>
  </docspec>
  <table name="VBAK">
    <pkfield name="VBELN" fktablename="Orders"
                          fkfieldname="VBELN"/>
    <fieldtoread name="VTWEG">
      <textref tablename="TVTWT" reffieldname="VTWEG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VKORG">
      <textref tablename="TVKOT" reffieldname="VKORG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VDATU"/>
    <fieldtoread name="VKBUR">
      <textref tablename="TVKBT" reffieldname="VKBUR"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VKGRP">
      <textref tablename="TVGRT" reffieldname="VKGRP"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VBTYP"/>
    <fieldtoread name="AUART"/>
  </table>
  <table name="MARA">
    <pkfield name="MATNR" fktablename="Orders"
                             fkfieldname="MATNR"/>
    <fieldtoread name="MATNR"/>
    <fieldtoread name="MTART">
      <textref tablename="T134T" reffieldname="MTART"
          textfieldname="MTBEZ" langfieldname="SPRAS"/>
    </fieldtoread>
  </table>
 </configuration>
</xmlextractor_tableconfiguration>
```

The **SD_C** table configuration contains the following tables:

DOCUMENT HEADER TABLE

```
...
  <docreftable name="VBAK">
   <booleancondition>
     <condition fieldname="ERDAT#-#ERZET"
         logicaloperator="creationtimestamp">
       <value>yyyyMMdd</value>
       <value>HHmmss</value>
     </condition>
     <condition fieldname="VBTYP"
                     logicaloperator="eq">
       <value>C</value>
     </condition>
   </booleancondition>
```

```
    <pkfield name="VBELN" />
  </docreftable>
...
```

The identifier and name of the document header table from the R/3 system is **VBAK**. All order document headers (**VBTYP=C**) for the period specified in the command line are read. The **value** elements of the **creationtimestamp** condition operator specify the format of the specified time stamp.

The name of the primary key table column is **VBELN**. For each different **VBELN** field value, data records are extracted from the linked document table for which **VBELN** has the same value as in the document header table.

DOCUMENT TABLE

```
...
  <doctable name="Orders" tablename="VBAP">
    <pkfield name="VBELN" fktablename="VBAK"
                      fkfieldname="VBELN"/>
    <pkfield name="POSNR" />
    <fieldtoread name="ERDAT"/>
    <fieldtoread name="ERZET"/>
    <fieldtoread name="MATNR">
      <textref tablename="MAKT" reffieldname="MATNR"
        textfieldname="MAKTX" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="KONDM">
      <textref tablename="T178T" reffieldname="KONDM"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="SPART">
      <textref tablename="TSPAT" reffieldname="SPART"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="WERKS">
      <textref tablename="T001W" reffieldname="WERKS"
                        textfieldname="NAME1"/>
    </fieldtoread>
    <fieldtoread name="CHARG"/>
    <fieldtoread name="PSTYV"/>
    <fieldtoread name="ERNAM"/>
    <fieldtoread name="NETWR"/>
  </doctable>
...
```

The document table **VBAP** is assigned the identifier **Orders**. The foreign key relation specified in the line

```
<pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
```

links the table to the document header table. Unless otherwise specified, foreign key relations are created based on equal field values (see chapter **Comparison of foreign key relations** (page 23)). In this example, only those table rows for which the comparison of the

VBAP-VBELN and **VBAK-VBELN** field values returns equal values will be extracted. The primary key consists of the **VBELN** and **POSNR** columns.

For each table row extracted, a system event (**event** XML element) is created in the XML output file. For each **fieldtoread** XML element, a line of the form

```
<attribute type="...">...</attribute>
```

is written to the output file.

The primary key fields (**pkfield**) specified in the **doctable** definition are extracted automatically. You do not have to specify **fieldtoread** elements for them.

For some **fieldtoread** elements, the value extracted from the referenced table (**textref** XML element) is written in addition to the extracted data field value. The optional **langfieldname** extracts the language-specific text of the data field.
The complete source system attribute type is made up of the identifier of the document header table (**doctable name**), the field name (**fieldtoread name**) and the name of the referenced text field (**textref ... textfieldname**). The source system attribute type for the first extracted **fieldtoread** element with a referenced table looks like this:

<attribute type="Orders-MATNR-MAKTX">...</attribute>

DATA TABLES

```
...
   <table name="VBAK">
     <pkfield name="VBELN" fktablename="Orders"
                           fkfieldname="VBELN"/>
     <fieldtoread name="VTWEG">
       <textref tablename="TVTWT" reffieldname="VTWEG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
     </fieldtoread>
     <fieldtoread name="VKORG">
       <textref tablename="TVKOT" reffieldname="VKORG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
     </fieldtoread>
     <fieldtoread name="VDATU"/>
     <fieldtoread name="VKBUR">
       <textref tablename="TVKBT" reffieldname="VKBUR"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
     </fieldtoread>
     <fieldtoread name="VKGRP">
       <textref tablename="TVGRT" reffieldname="VKGRP"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
     </fieldtoread>
     <fieldtoread name="VBTYP"/>
     <fieldtoread name="AUART"/>
   </table>
```

```
   <table name="MARA">
     <pkfield name="MATNR" fktablename="Orders"
                              fkfieldname="MATNR"/>
     <fieldtoread name="MATNR"/>
     <fieldtoread name="MTART">
       <textref tablename="T134T" reffieldname="MTART"
          textfieldname="MTBEZ" langfieldname="SPRAS"/>
     </fieldtoread>
   </table>
...
```

The foreign key relations to the **VBELN** and **MATNR** primary key fields in the **Orders** table are used to extract supplementary information from the **VBAK** and **MARA** data tables.

4.3.9.2 XML output file (PPM system event format)

The file extract below shows a range of system events from the XML output file generated using the configuration file specified in the chapter on **Table configuration** (page 31):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="Orders-CHARG"></attribute>
  <attribute type="Orders-ERDAT">20000214</attribute>
  <attribute type="Orders-ERNAM">HDM</attribute>
  <attribute type="Orders-ERZET">143616</attribute>
  <attribute type="Orders-KONDM"></attribute>
  <attribute type="Orders-MATNR">P-100</attribute>
  <attribute type="Orders-MATNR-MAKTX">
     Thread rod M'8 DIN '8895' without tolerance
  </attribute>
  <attribute type="Orders-NETWR">28.70</attribute>
  <attribute type="Orders-POSNR">000010</attribute>
  <attribute type="Orders-PSTYV">TAN</attribute>
  <attribute type="Orders-SPART">01</attribute>
  <attribute type="Orders-SPART-VTEXT">Product category 01
  </attribute>
  <attribute type="Orders-VBELN">0000000001</attribute>
  <attribute type="Orders-WERKS">1000</attribute>
  <attribute type="Orders-WERKS-NAME1">Plant 1000 (Hamburg)
  </attribute>
  <attribute type="MARA-MATNR">P-100</attribute>
  <attribute type="MARA-MTART">HAWA</attribute>
  <attribute type="MARA-MTART-MTBEZ">Trading goods
  </attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="VBAK-VKBUR"></attribute>
  <attribute type="VBAK-VKGRP"></attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
```

```
  </attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
  </attribute>
</event>
<event>
  <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
  </attribute>
  <attribute type="Orders-SPART">01</attribute>
  <attribute type="Orders-PSTYV">TAD</attribute>
  <attribute type="Orders-MATNR">SERVICE</attribute>
  <attribute type="Orders-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR"></attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Orders-SPART-VTEXT">Product category 01
  </attribute>
  <attribute type="Orders-ERZET">143616</attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="Orders-POSNR">000020</attribute>
  <attribute type="MARA-MTART-MTBEZ">Service
  </attribute>
  <attribute type="Orders-WERKS-NAME1">Plant 1000 (Hamburg)
  </attribute>
  <attribute type="Orders-CHARG"></attribute>
  <attribute type="Orders-WERKS">1000</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="Orders-MATNR-MAKTX">Repair
  </attribute>
  <attribute type="Orders-VBELN">0000000001</attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VKGRP"></attribute>
  <attribute type="Orders-NETWR">179.00</attribute>
  <attribute type="MARA-MTART">DIEN</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="Orders-ERNAM">HDM</attribute>
  <attribute type="MARA-MATNR">SERVICE</attribute>
  <attribute type="Orders-KONDM"></attribute>
</event>
<event>
  <attribute type="VBAK-VKGRP-BEZEI">GR. F2 Mr. Mayer
  </attribute>
  <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
  </attribute>
  <attribute type="Orders-SPART">01</attribute>
  <attribute type="Orders-PSTYV">TAN</attribute>
  <attribute type="Orders-MATNR">P-100</attribute>
  <attribute type="Orders-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR">1000</attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Orders-SPART-VTEXT">Product category 01
  </attribute>
  <attribute type="Orders-ERZET">131257</attribute>
```

```
  <attribute type="VBAK-VKBUR-BEZEI">Frankfurt office
  </attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="Orders-POSNR">000010</attribute>
  <attribute type="MARA-MTART-MTBEZ">Trading goods</attribute>
  <attribute type="Orders-WERKS-NAME1">Plant 1000 (Hamburg)
  </attribute>
  <attribute type="Orders-CHARG"></attribute>
  <attribute type="Orders-WERKS">1000</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="Orders-MATNR-MAKTX">
     Thread rod M'8 DIN '8895' without tolerance
  </attribute>
  <attribute type="Orders-VBELN">0000000002</attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VKGRP">101</attribute>
  <attribute type="Orders-NETWR">28.70</attribute>
  <attribute type="MARA-MTART">HAWA</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="Orders-ERNAM">HDM</attribute>
  <attribute type="MARA-MATNR">P-100</attribute>
  <attribute type="Orders-KONDM"></attribute>
</event>
...
</eventlist>
```

4.3.10 Block extraction

You can influence the relationship between the execution and memory efficiency of the extraction operation.
The value of the **-cpd** command line argument determines the number of system events that are simultaneously held in the system memory. The maximum possible number depends on the size of the available system memory.

The actual system memory used depends on the following factors:

- Total number of system events to be extracted in the defined analysis period
- Number of system events held in parallel in the system (concurrently processed documents)
- Number of data fields to be extracted (**fieldtoread** and **reftext** XML elements).
- Memory requirements of extracted data fields

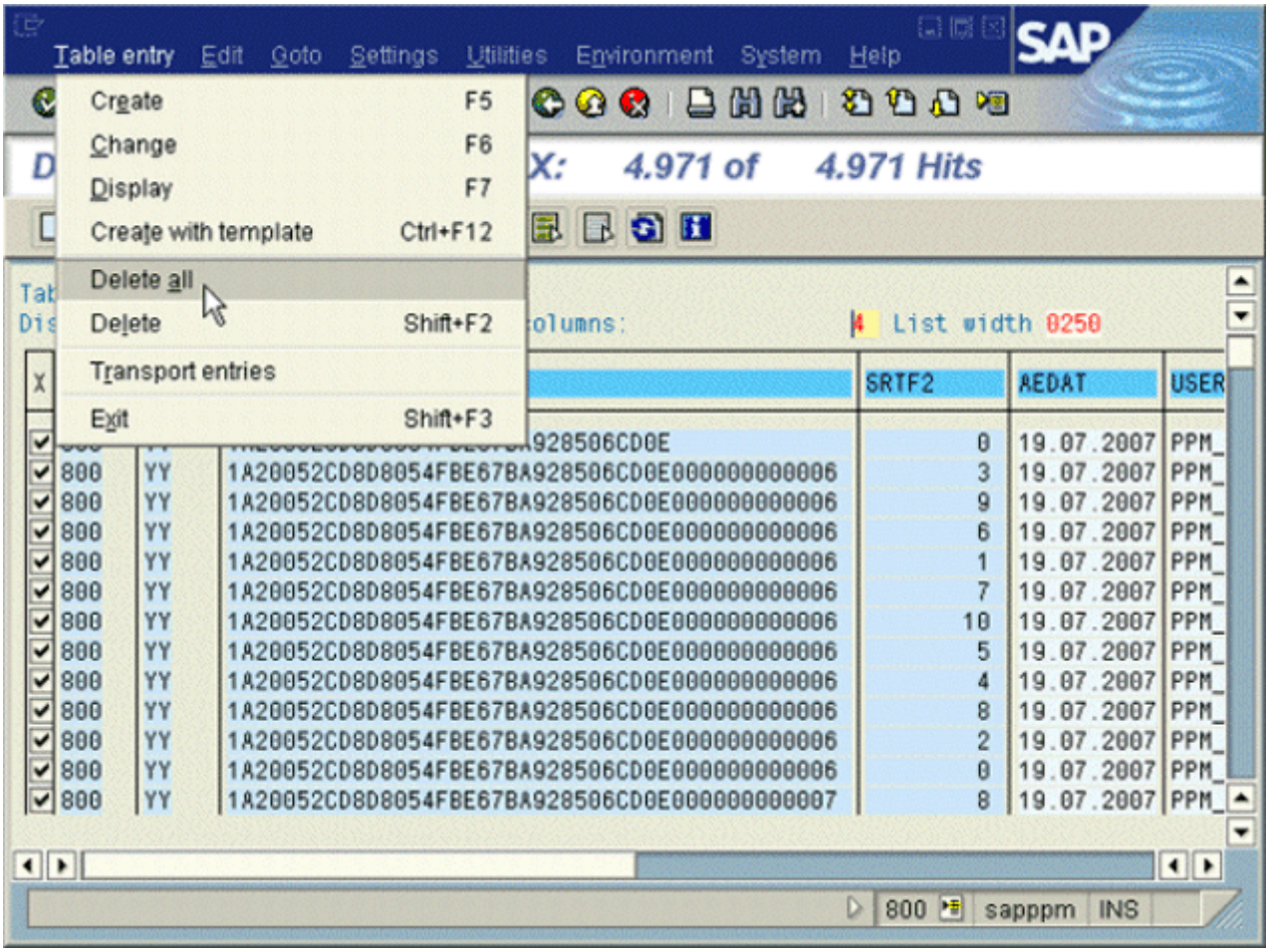
4.3.10.1 Block extraction of large data volumes

During the extraction, all data of an RFC call is transferred at once to the Java client using SAP JCo. However, large data volumes can result in memory overflow (OutOfMemoryError).

For example, this happens when the primary keys for the document tables and document header tables are being determined. In this case, the data must first be buffered in the R/3 system and then transferred to the Java client a block at a time. The block size is specified by the **-cpd** parameter. There are three different block modes, which differ in terms of the way in which the data is buffered in the R/3 system.

BLOCK MODE D

Mode **D** is recommended for block extraction of large data sets from the R/3 system. The selected data is stored in the **/IDS/INDX** database table. If the connection to the Extractor fails or the extraction operation is aborted manually, the data is not automatically deleted from the database table. You need to delete the buffered data manually. To do this, display the content of the **/IDS/INDX** table using transaction **SE16**. Then select all the relevant entries and delete them using the **Delete all...** option in the main **Table entry** menu.



Example

You want to extract more than a million entries from the **CPDOS** table (change document items). If you do not use block extraction, the Java memory will overflow and extraction will be aborted.

Executing the following command line extracts the data in blocks (block parameter in bold):

```
runsap2ppm -systemconfig Systemconfig.xml -tableconfig TableconfigCDPOS.xml
-outfile eventsCDPOS -nozip -begindate 19700101 -rfc_blockmode D -cpd 10000
```

By specifying the **-rfc_blockmode** parameter with the **D** argument, you start block extraction, which writes all data from the **CPDOS** database table to the output file. You use the **-cpd** parameter to specify the size of the blocks to be processed. In the example, **10000** documents are processed simultaneously. The default value is **5000**.

The progress of the block extraction is logged as specified, for example:

```
I: 09.05.06 11:30:38: [XML] Extracting data from...
...
...
I: 09.05.06 11:32:30: [XML]    "CDPOS" table:
       10,000 of 1,036,995 rows extracted.
I: 09.05.06 11:32:30: [XML] Extracting attributes for
                        10000 source system events...
I: 09.05.06 11:32:39: [XML]    "CDPOS" table:
       20,000 of 1,036,995 rows extracted.
I: 09.05.06 11:32:39: [XML] Extracting attributes for
                        10000 source system events...
...
```

By specifying **-progress no**, you can suppress the additional log messages for the block modes, which show the progress of the block extraction.

BLOCK MODE G

The selected data is stored in an internal table of the function group. The data is held in the R/3 server memory. If the connection to the Extractor fails or the extraction is aborted manually, the data is automatically deleted in the R/3 system. These specifications apply from SAP Release 4.7, that is, from SAP Kernel Release 6.10.

BLOCK MODE M

The selected data is stored in the ABAP memory using **export to memory id**. The data is held in the R/3 server memory. If the connection to the Extractor fails or the extraction is aborted manually, the data is automatically deleted in the R/3 system. These specifications apply from SAP Release 4.7, that is, from SAP Kernel Release 6.10.

4.3.11 Extract change documents

Change documents represent a special document type in the SAP R/3 system. These documents are stored in the **CDHDR** (change header) and **CDPOS** (change item) tables. As these tables are structured in the same way as the other document header and document tables in the R/3 system, the configuration structure can be used to extract data from the change document tables with no changes.

The configuration for extracting data from the change document tables must be designed in such a way that data is extracted from the **CDHDR** change header table before any other tables.

There are two ways to extract change documents.

4.3.11.1 Create source system events from individual changes

A system event is created for each change document.

Example

The changes for January 2003 that meet the following conditions are to be extracted from the change document table:

- The object class is **VERKBELEG**.
- The changed table is **VBAK**.
- The changed field is **VKGRP**.
- The new field value is **4711** or **4712**.

The conditions can be formulated in the XML configuration file below:

```
...
<configuration name="OneEventForEveryChangeDoc">
  <docspec>
    <docreftable name="CDHDR">
      <booleancondition logicaloperator="AND">
        <condition fieldname="OBJECTCLAS"
                              logicaloperator="eq">
          <value>VERKBELEG</value>
        </condition>
        <condition fieldname="UDATE#-#UTIME"
                  logicaloperator="creationtimestamp">
          <value>yyyyMMdd</value>
          <value>HHmmss</value>
        </condition>
      </booleancondition>
      <pkfield name="OBJECTCLAS"/>
      <pkfield name="OBJECTID"/>
      <pkfield name="CHANGENR"/>
    </docreftable>

    <doctable name="CDPOS">
      <booleancondition logicaloperator="AND">
        <condition fieldname="TABNAME"
                              logicaloperator="eq">
          <value>VBAK</value>
        </condition>
        <condition fieldname="FNAME"
                              logicaloperator="eq">
          <value>VKGRP</value>
```

```
      </condition>
      <condition fieldname="VALUE_NEW"
                           logicaloperator="in">
        <value>4711</value>
        <value>4712</value>
      </condition>
    </booleancondition>
    <pkfield name="OBJECTCLAS" fktablename="CDHDR"
                        fkfieldname="OBJECTCLAS"/>
    <pkfield name="OBJECTID" fktablename="CDHDR"
                         fkfieldname="OBJECTID"/>
    <pkfield name="CHANGENR" fktablename="CDHDR"
                         fkfieldname="CHANGENR"/>
    <pkfield name="TABNAME"/>
    <pkfield name="TABKEY"/>
    <pkfield name="FNAME"/>
    <pkfield name="CHNGIND"/>
    <fieldtoread name="TABNAME"/>
    <fieldtoread name="CHNGIND"/>
    <fieldtoread name="VALUE_NEW"/>
    <fieldtoread name="VALUE_OLD"/>
  </doctable>
 </docspec>

 <table name="CDHDR">
    <pkfield name="OBJECTCLAS" fktablename="CDPOS"
                        fkfieldname="OBJECTCLAS"/>
    <pkfield name="OBJECTID" fktablename="CDPOS"
                         fkfieldname="OBJECTID"/>
    <pkfield name="CHANGENR" fktablename="CDPOS"
                         fkfieldname="CHANGENR"/>
    <fieldtoread name="UDATE"/>
    <fieldtoread name="UTIME"/>
    <fieldtoread name="CHANGENR"/>
    <fieldtoread name="USERNAME"/>
  </table>
</configuration>
...
```

4.3.11.2 Extract system events including changes

Information on change documents from the SAP tables **CDPOS** and **CDHDR** is stored in special source system attributes of system events using a particular class.

Change documents are extracted using a suitable configuration of a data table. The **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZChangeDocTable_sap2ppm** class is used to extract the change documents.

If there are no change documents for an extracted system events, only the configured table fields are transferred as source system attributes.

In the configuration of the data table from which the change documents are to be extracted, the following conditions must be specified:

```
<table name="..." classtouse="com.idsscheer.ppm.
          xmlextractortools.extractor.sap2ppm.
                   ZChangeDocTable_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" ...>
      ...
    </condition>
    <condition fieldname="OBJECTID" ...>
      ...
    </condition>
    <condition fieldname="TABNAME" ...>
      ...
    </condition>
    <booleancondition logicaloperator="OR">
      <condition fieldname="TABKEY" ...>
        ...
      </condition>
      <condition fieldname="TABKEY" ...>
        ...
      </condition>
      ...
    </booleancondition>
  </booleancondition>
  ...
</table>
```

Please do not specify any other conditions.

The following table displays the source system attributes that are created from the extracted change document information. **CH_DOCS** has been specified as the identifier of the data table in the configuration.

Source system attribute	Description
CH_DOCS-SUM_OF_CHANGES	Total number of changes in the extracted document (number of saving operations).
CH_DOCS-FIRST_CHANGE_DATE CH_DOCS-FIRST_CHANGE_TIME	Date of the first change Time of the first change
CH_DOCS-LAST_CHANGE_DATE CH_DOCS-LAST_CHANGE_TIME	Date of the last change Time of the last change

Source system attribute	Description
CH_DOCS-CHANGE_USER_<x>	Name of the processor that has created change documents. The attribute is created for every processor.**<x>** is the consecutive number of the processor.
CH_DOCS-NUM_OF_CHANGES_USER_<x>	Total number of changes made by the processor.**<x>** is the consecutive number of the processor.

Example

The file extract below illustrates the configuration for extracting the changes in the **VBAK** and **VBAP** tables:

```
<configuration name="SD_CHANGE_DOC_SUM">
  <docspec>
    <docreftable name="VBAK">
      <condition fieldname="ERDAT#-#ERZET"
                 logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <pkfield name="VBELN" />
    </docreftable>
    <doctable name="VBAP">
      <pkfield name="VBELN" fktablename="VBAK"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" />
      <fieldtoread name="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="ERDAT"/>
      <fieldtoread name="ERZET"/>
      <fieldtoread name="ERNAM"/>
    </doctable>
  </docspec>
  <table name="CH_DOCS" classtouse="com.idsscheer.ppm.
                        xmlextractortools.extractor.
                    sap2ppm.ZChangeDocTable_sap2ppm">
    <booleancondition>
      <condition fieldname="OBJECTCLAS">
        <value>VERKBELEG</value>
      </condition>
      <condition fieldname="OBJECTID"
              logicaloperator="eq_concatEventAttrValues">
        <value>VBAP-VBELN</value>
      </condition>
      <condition fieldname="TABNAME" logicaloperator="in">
        <value>VBAK</value>
        <value>VBAP</value>
      </condition>
```

```
      <booleancondition logicaloperator="OR">
        <condition fieldname="TABKEY"
              logicaloperator="eq_concatEventAttrValues">
          <value>CLIENT#-#</value>
          <value>VBAP-VBELN#-#</value>
          <value>VBAP-POSNR</value>
        </condition>
        <condition fieldname="TABKEY"
              logicaloperator="eq_concatEventAttrValues">
          <value>CLIENT#-#</value>
          <value>VBAP-VBELN</value>
        </condition>
      </booleancondition>
    </booleancondition>
    <pkfield name="VBELN" fktablename="VBAP"
                                fkfieldname="VBELN"/>
    <pkfield name="POSNR" fktablename="VBAP"
                                fkfieldname="POSNR"/>
  </table>
</configuration>
```

The **eq_concatEventAttrValues** operator compares the content of the specified field with combined attribute values from the specified fields. The values specified in the **value** XML elements are interpreted as the identifiers of source system attributes already extracted.

Note on the values of the OBJECTID and TABKEY fields

The individual field values (<value>...</value>) are concatenated using the specified separator string **#-#** between the individual field values. In the example, the values of the fields **OBJECTID** and **TABKEY** contain the following:

OBJECTID

Value	Description	Example value
VBAP-VBELN	The value already extracted from the field **VBELN** in the table **VBAP** must match as a condition for the field **OBJECTID**.	0000004972

TABKEY

Value	Description	Example value
CLIENT	Parameter that is replaced by the number of the SAP client from which you extract.	800

Value	Description	Example value
VBAP-VBELN	Value of the extracted field **VBELN** of the table **VBAP**.	0000004972
VBAP-POSNR	Value of the extracted field **POSNR** of the table **VBAP**.	000020

Based on the example values in the above configuration, the concatenated values 800#-#0000004972#-#000020 and 800#-#0000004972 are the result for the field **TABKEY** (key of the changed table row).

The output file extract below illustrates a possible system event:

```
<event>
  <attribute type="CH_DOCS-CHANGE_USER_1">
      sapuser
  </attribute>
  <attribute type="CH_DOCS-CHANGE_USER_2">
      ANFEL
  </attribute>
  <attribute type="CH_DOCS-FIRST_CHANGE_DATE">
      20030919
  </attribute>
  <attribute type="CH_DOCS-FIRST_CHANGE_TIME">
      181020
  </attribute>
  <attribute type="CH_DOCS-LAST_CHANGE_DATE">
      20030930
  </attribute>
  <attribute type="CH_DOCS-NUM_OF_CHANGES_USER_1">
      1
  </attribute>
  <attribute type="CH_DOCS-NUM_OF_CHANGES_USER_2">
      3
  </attribute>
  <attribute type="CH_DOCS-SUM_OF_CHANGES">
      4
  </attribute>
  <attribute type="VBAP-ERDAT">20030919</attribute>
  <attribute type="VBAP-ERNAM">ANFEL</attribute>
  <attribute type="VBAP-ERZET">174938</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
  <attribute type="VBAP-VBELN">0000007500</attribute>
</event>
```

4.3.11.3 Extract the first change document of a table field

Use the class **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZFirstFieldChange_sap2ppm** to extract information on the first change of a table field from an SAP system. Specify the class with the XML attribute **classtouse** in the table configuration in the XML element **table**.

CALCULATION

For some documents in the SAP system, there is only one field for the last change date. The initial value of this field in the first change document is interpreted as the creation date of the document. This value is extracted with the class to be used from the change document tables **CDHDR** and **CDPOS**.

THE CONDITIONS FOR USING THE CLASS MUST BE AS FOLLOWS:

```
<table name="name" classtouse="com.idsscheer.ppm.
                          xmlextractortools.
                          extractor.sap2ppm.
                          ZFirstFieldChange_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" ...>
      ...
    </condition>
    <condition fieldname="OBJECTID" ...>
      ...
    </condition>
    <condition fieldname="TABNAME" ...>
      ...
    </condition>
    <condition fieldname="TABKEY" ...>
      ...
    </condition>
    <condition fieldname="FNAME" ...>
      ...
    </condition>
  </booleancondition>

  <pkfield name="OBJECTCLAS"/>
  <pkfield name="OBJECTID"/>
  <pkfield name="CHANGENR"/>
  <pkfield name="TABNAME"/>
  <pkfield name="TABKEY"/>
  <pkfield name="FNAME"/>
  <pkfield name="CHNGIND"/>

  <pkfield name="<Field that is used in the tabkey>"
     fktablename="<Foreign key table, from
                  which the field is extracted>"
     fkfieldname="<Foreign key table field>"/>
  ... Other fields used in the tabkey
  <fieldtoread name="CDPOS-VALUE_OLD"/>
  ... Other fields from the CDPOS table
```

```
  <fieldtoread name="CDHDR-UDATE"/>
  ... Other fields from the CDHDR table
</table>
```

The combination of values specified in the **OBJECTID** and **TABKEY** fields is the same as for the **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ↵ ZChangeDocTable_sap2ppm** (see chapter **Extract system events incl. changes** (page 41)).

Example (of an SAP configuration)

With this configuration, you want to extract data as to who last changed the field containing the approval status of purchase requisitions and when:

```
<table name="ORIGINAL_ERDAT" classtouse="com.idsscheer.ppm. ↵
                  xmlextractortools.extractor.sap2ppm. ↵
                        ZFirstFieldChange_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" logicaloperator="eq">
      <value>EINKBELEG</value>
    </condition>
    <condition fieldname="OBJECTID" ↵
              logicaloperator="eq_concatEventAttrValues">
      <value>EKPO-EBELN</value>
    </condition>
    <condition fieldname="TABNAME" logicaloperator="eq">
      <value>EKPO</value>
    </condition>
    <condition fieldname="TABKEY" ↵
            logicaloperator="eq_concatEventAttrValues">
      <value>CLIENT#-#EKPO-EBELN#-#EKPO-EBELP</value>
    </condition>
    <condition fieldname="FNAME" logicaloperator="eq">
      <value>AEDAT</value>
    </condition>
  </booleancondition>

  <pkfield name="OBJECTCLAS"/>
  <pkfield name="OBJECTID"/>
  <pkfield name="CHANGENR"/>
  <pkfield name="TABNAME"/>
  <pkfield name="TABKEY"/>
  <pkfield name="FNAME"/>
  <pkfield name="CHNGIND"/>

  <pkfield name="EBELN" fktablename="EKPO" ↵
                    fkfieldname="EBELN"/>
  <pkfield name="EBELP" fktablename="EKPO" ↵
                    fkfieldname="EBELP"/>

  <fieldtoread name="CDPOS-VALUE_OLD"/>
</table>
```

The result of the value extraction with this table configuration could be as follows in a created system event:

```
<event>
  ...
  <attribute type="ORIGINAL_ERDAT-VALUE_OLD">
    20010402
  </attribute>
  ...
</event>
```

4.3.11.4 Extract the last change document of a table field

Use the class **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.** ↵ **ZLastFieldChange_sap2ppm** to extract information on the last change of a document from an SAP system. Configuration and calculation of the class are the same as for the class **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.** ↵ **ZFirstFieldChange_sap2ppm** (see chapter **Extract the first change document of a table field** (page 46)), only with the difference that the last change document instead of the first is used for determining the data.

4.3.11.5 Extract the last change document from multiple table fields

Information about change documents is stored in the **CDHDR** (change document header) and **CDPOS** (change document items) change document tables. This information can be extracted by carrying out the corresponding configuration for the relevant data table (**table** XML element). Please use the **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ZChangeDocTableNew_sap2ppm** class to do this.

Example (extract from table configuration, general file framework)

A configuration with conditions that need to be specified in a **table** element could look like this:

```
...
<table name="tablename" classtouse="com.idsscheer.ppm.
               xmlextractortools.extractor.sap2ppm.
                          ZChangeDocTableNew_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="OBJECTID" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="TABNAME" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="TABKEY" logicaloperator="...">
```

```
      ...
    </condition>
    <condition fieldname="FNAME" logicaloperator="in">
      ...
    </condition>
</booleancondition>

<pkfield name="OBJECTCLAS"/>
<pkfield name="OBJECTID"/>
<pkfield name="CHANGENR"/>
<pkfield name="TABNAME"/>
<pkfield name="TABKEY"/>
<pkfield name="FNAME"/>
<pkfield name="CHNGIND"/>

<pkfield name="<Field that is used in TABKEY>"
        fktablename="<Foreign key table, from
                  which the field is extracted>"
        fkfieldname="<Foreign key table field>"/>
... Other fields used in TABKEY
<fieldtoread name="CDPOS-VALUE_OLD"/>
... Other fields from the CDPOS table
<fieldtoread name="CDHDR-UDATE"/>
... Other fields from the CDHDR table
</table>
...
```

The combination of values specified in the **OBJECTID** and **TABKEY** fields is the same as for the **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZChangeDocTable_sap2ppm** class. A more concrete example is shown below.

Example (extract from a table configuration from the PPM4MM configuration package)

The example configuration is intended to extract the time at which one of the fields for the reasons for blocking logistical invoices was changed:

```
...
  <table name="CDPOS" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.sap2ppm.
                           ZChangeDocTableNew_sap2ppm">
    <booleancondition>
      <condition fieldname="OBJECTCLAS" logicaloperator="eq">
        <value>INCOMINGINVOICE</value>
      </condition>
      <condition fieldname="OBJECTID"
                 logicaloperator="eq_concatEventAttrValues">
        <value>RSEG-BELNR#-#RSEG-GJAHR</value>
      </condition>
      <condition fieldname="TABNAME" logicaloperator="eq">
        <value>RSEG</value>
      </condition>
      <condition fieldname="TABKEY"
                 logicaloperator="eq_concatEventAttrValues">
        <value>   #-#RSEG-BELNR#-#
                          RSEG-GJAHR#-#RSEG-BUZEI</value>
      </condition>
```

```
      <condition fieldname="FNAME" logicaloperator="in">
        <value>SPGRP</value>
        <value>SPGRM</value>
        <value>SPGRT</value>
        <value>SPGRG</value>
        <value>SPGRV</value>
        <value>SPGRQ</value>
        <value>SPGRS</value>
        <value>SPGRC</value>
      </condition>
    </booleancondition>
    <pkfield name="OBJECTCLAS"/>
    <pkfield name="OBJECTID"/>
    <pkfield name="CHANGENR"/>
    <pkfield name="TABNAME"/>
    <pkfield name="TABKEY"/>
    <pkfield name="FNAME"/>
    <pkfield name="CHNGIND"/>
    <pkfield name="BELNR" fktablename="RSEG"
                                fkfieldname="BELNR"/>
    <pkfield name="GJAHR" fktablename="RSEG"
                                fkfieldname="GJAHR"/>
    <pkfield name="BUZEI" fktablename="RSEG"
                                fkfieldname="BUZEI"/>
  </table>
...
```

A system event generated with this example configuration could look like this:

```
...
<event>
  <attribute type="CDPOS-CHANGE_USER_1">HUETT</attribute>
  <attribute type="CDPOS-FIRST_CHANGE_DATE">20010220
  </attribute>
  <attribute type="CDPOS-FIRST_CHANGE_TIME">154943
  </attribute>
  <attribute type="CDPOS-LAST_CHANGE_DATE">20010220
  </attribute>
  <attribute type="CDPOS-LAST_CHANGE_TIME">154943
  </attribute>
  <attribute type="CDPOS-NUM_OF_CHANGES_USER_1">1
  </attribute>
  <attribute type="CDPOS-SUM_OF_CHANGES">1</attribute>
  ...
<event>
...
```

4.3.12 Extract the first or last line in a sorting

The **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZSortWithInteger_sap2ppm** Java class to be used extracts data records from tables, sorts them using the specified sorting criteria, and writes either the first

or last data record to the system events to be generated. The class must be specified in the **classtouse** attribute of the **table** XML element.

Example

Status history documents represent a special document type in the SAP R/3 system. These documents are stored in the **JCDS** table. The class is to determine the last change relating to a particular status type and to write this change to the corresponding system events. To do this, all order numbers are extracted from the **AUFK** table. In addition, for each order number with status information, all entries for that order number with the **OR** prefix and the status **I0043** (**STAT** field) are extracted from the **JCDS** table. These are to be sorted by creation time (**UDATE**, **UTIME**) and the last data record created is to be extracted. All field values pertaining to this data record are to be written to the relevant system event.

The file extract below illustrates the configuration for extracting status information from the **JCDS** table under the above conditions:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
             'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="AUFK_JCDS">
    <docspec>
      <doctable name="AUFK">
        <pkfield name="AUFNR"/>
      </doctable>
    </docspec>
    <table name="JCDS_I0043" tablename="JCDS"
             classtouse="com.idsscheer.ppm.
               xmlextractortools.extractor.
           sap2ppm.ZSortWithInteger_sap2ppm">
      <parameter name="SORTCRITERION">
        <value>UDATE</value>
        <value>UTIME</value>
      </parameter>
      <parameter name="USE">
        <value>MAX</value>
      </parameter>
      <condition fieldname="STAT"
                          logicaloperator="eq">
        <value>I0043</value>
      </condition>
      <pkfield name="OBJNR" fktablename="AUFK"
                           fkfieldname="AUFNR">
        <prefix>
          <value>OR</value>
        </prefix>
      </pkfield>
      <fieldtoread name="CDTCODE"/>
      <fieldtoread name="CHGNR"/>
      <fieldtoread name="CHIND"/>
      <fieldtoread name="INACT"/>
      <fieldtoread name="OBJNR"/>
      <fieldtoread name="STAT"/>
```

```
      <fieldtoread name="TCODE"/>
      <fieldtoread name="USNAM"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

A system event extracted with this configuration could look like this:

```
<event>
  <attribute type="AUFK-AUFNR">000000800002</attribute>
  <attribute type="JCDS_I0043-CDTCODE"></attribute>
  <attribute type="JCDS_I0043-CHGNR">002</attribute>
  <attribute type="JCDS_I0043-CHIND">U</attribute>
  <attribute type="JCDS_I0043-INACT">X</attribute>
  <attribute type="JCDS_I0043-OBJNR">
                  OR000000800002
  </attribute>
  <attribute type="JCDS_I0043-STAT">I0043</attribute>
  <attribute type="JCDS_I0043-TCODE">KOK2</attribute>
  <attribute type="JCDS_I0043-UDATE">19961217</attribute>
  <attribute type="JCDS_I0043-USNAM">MUELLERJ</attribute>
  <attribute type="JCDS_I0043-UTIME">192122</attribute>
</event>
```

The sorting criteria (in this case **UDATE**, **UTIME**) are automatically included in the system event specification.

The template for creating the configuration shown looks like this (the XML elements and attributes that are optional is outlined in the table below):

```
<table name="..." tablename="..." classtouse=
                      "com.idsscheer.ppm.
              xmlextractortools.extractor.
          sap2ppm.ZSortWithInteger_sap2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
    <value>...</value>
    ...
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ...
  See chapter
  R3 table configuration (system event specification) (page 11)
  ...
</table>
```

The table below reiterates the most important configuration entries for the above **table** definition:

XML element/attribute	Value: Description
name	The specified name is added to the extracted source system attributes as a prefix

XML element/attribute	Value: Description
tablename (optional)	Table from which the information is to be extracted. Default value: Value from **name** XML attribute
classtouse	**com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZSortWithInteger_sap2ppm**: Java class to be used
parameter name	
SORTCRITERION	First parameter to be specified. At least one **value** XML element must be specified (field name of sorting criterion). The specified fields may only contain integer values. If several sorting criteria are specified, they are prioritized from top to bottom, that is, the data records are first compared against the first (top) criterion and then against the second criterion if no complete sorting has been created, and so on.
USE	Second parameter to be specified. A single **value** XML element must be specified. Valid values: **MIN** (the data record with the lowest integer value for the sorting criterion is selected) **MAX** (the data record with the highest integer value for the sorting criterion is selected) Either the first or last data record from the list generated based on the specified sorting criteria is selected to be extracted.

You should use this class for extracting the first or last line of a sorting sparingly, as the sorting and selection operations result in a loss of performance and memory.

4.3.13 Create attributes with inverted date

You can use the Java class **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableInvertDates_sap2ppm** to create new system event attributes with

inverted dates when extracting data from an SAP system. Values of already extracted date attributes will be inverted. An inverted date is calculated by extracting the value of the date in SAP format, for example, 20110529 for July 29, 2011 from 99999999. The new system events created can be referenced in other data table configurations (**table** XML element).

Example

```
<docspec>
    <doctable name="VBAP">
      <pkfield name="VBELN" />
      <pkfield name="POSNR" />
      <pkfield name="ERDAT"/>
      <pkfield name="AEDAT"/>
      <pkfield name="ABDAT"/>
    </doctable>
</docspec>
<table name="VBAP_DATES_INVERTED" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableI
nvertDates_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP" fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP" fkfieldname="AEDAT" />
</table>
```

If you are using the **ZTableInvertDates_sap2ppm** class, you do not need to specify a **fieldtoread** element. Any specified **fieldtoread** elements will be ignored. The new system event attributes consist of the value of the **name** attribute of the **table** element and the value of the **name** attribute of the **pkfield** element. In the example, two new system event attributes are created, **VBAP_DATES_INVERTED-ERDAT** and **VBAP_DATES_INVERTED-AEDAT**.

You can use the regular **fkpart**, **prefix**, and **postfix** operations for the **pkfield** elements.

Example

A system event created with the above configuration could look like this:

```
...
<event>
    <attribute type="VBAP-ABDAT">00000000</attribute>
    <attribute type="VBAP-AEDAT">19970127</attribute>
    <attribute type="VBAP-ERDAT">19970121</attribute>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004974</attribute>
    <attribute type="VBAP_DATES_INVERTED-AEDAT">80029872</attribute>
    <attribute type="VBAP_DATES_INVERTED-ERDAT">80029878</attribute>
</event>
...
```

You can use the class **ZTableInvertDates_sap2ppm** to convert an inverted date back to a normal date in SAP format.

Example

```
<docspec>
    <doctable name="VBAP">
      <pkfield name="VBELN" />
      <pkfield name="POSNR" />
      <pkfield name="ERDAT"/>
      <pkfield name="AEDAT"/>
      <pkfield name="ABDAT"/>
      </doctable>
</docspec>
<table name="VBAP_DATES_INVERTED" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableI
nvertDates_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP" fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP" fkfieldname="AEDAT" />
</table>
<table name="VBAP_DATES_INVERT_AGAIN" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableI
nvertDates_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP_DATES_INVERTED"
    fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP_DATES_INVERTED"
    fkfieldname="AEDAT" />
    <pkfield name="ABDAT" fktablename="VBAP_DATES_INVERTED"
    fkfieldname="ABDAT" />
</table>
```

Returns the following result:

```
...
<event>
    <attribute type="VBAP-ABDAT">00000000</attribute>
    <attribute type="VBAP-AEDAT">19970127</attribute>
    <attribute type="VBAP-ERDAT">19970121</attribute>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004974</attribute>
    <attribute type="VBAP_DATES_INVERTED-AEDAT">80029872</attribute>
    <attribute type="VBAP_DATES_INVERTED-ERDAT">80029878</attribute>
    <attribute type="VBAP_DATES_INVERT_AGAIN-AEDAT">19970127</attribute>
    <attribute type="VBAP_DATES_INVERT_AGAIN-ERDAT">19970121</attribute>
</event>
...
```

4.3.14 Multiplication of system events in tables

When extracting from an SAP system you have the option of creating additional system events based on a system event table.

At the source system level, a 1:n relationship may exist that cannot be resolved when determining the system events. Use the class **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableMultiplyEvents_sap2ppm** to create multiple system events from a system event and thus resolve the 1:n relationship.

Example

```
<configuration name="MultiplyEvents">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
         <value>0000006662</value>
         <value>0000006741</value>
       </condition>

       <pkfield name="VBELN" />
      </doctable>
    </docspec>

    <table name="VBAP"
    classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTa
    bleMultiplyEvents_sap2ppm">
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

    <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

This is the associated system event output file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1460 P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-06</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
</event>
```

```
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">MAG PA/DX 175</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-14</attribute>
  <attribute type="VBAP-POSNR">000030</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Jotachi SN4500</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-18</attribute>
  <attribute type="VBAP-POSNR">000040</attribute>
</event>
</eventlist>
```

The following example (excluding the class described) explains the procedure during extraction:

```
<configuration name="MultiplyEvents_Doctable_Only">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
        <value>0000006662</value>
        <value>0000006741</value>
       </condition>

       <pkfield name="VBELN" />
      </doctable>
    </docspec>

    <table name="VBAP" >
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

      <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Since multiple entries in the **VBAP** table are assigned to an entry in the **VBAK** table only one random row is extracted from the **VBAP** table.

This is the associated system event output file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
</eventlist>
```

4.3.15 Extract fields of individual tables for Data analytics

To enable easy data extraction for Data analytics, you can extract the entire contents of a source system table into a file in event format, which can then be imported into a Data analytics analysis realm.

You can also restrict the data to be extractes (page 60) by defining conditions for data extraction.

The table to be extracted is not configured using the table configuration but in the data source file itself. For this, the file **datasource.dtd** contains the following entries:

XML element/attribute	Description
analysistype	XML attribute: Must have the value **DATA_ANALYTICS**.
realmtable	Comprehensive XML element for configuring the Data analytics data source.
tablename	XML attribute of the **realmtable** element: Name of the table in the source system
sourcetable	Comprehensive XML element for configuring the Data analytics data source table. Must include at least one **sourcefield** element.
tablename	XML attribute of the **sourcetable** element: Name of the table in the source system
sourcefield	Contains the name of the field of the source system table.

The **analysistype** attribute of the XML element **datasource** must have the value **DATA_ANALYTICS** if the **<realmtable>** element specifies an analysis realm table (default value: PROCESS).

The **tablename** attribute of the **<realmtable>** element indicates the table name of the target table in the analysis realm configuration. The table name does not affect extraction itself, but is evaluated only by the PPM import.

Further information on data import for Data analytics is available in the PPM Data Analytics user guide.

The XML element **<realmtable>** contains the optional element **<sourcetable>** that specifies the table to be extracted. The columns to be extracted from this table must be specified in the **<sourcefield>** element. The element **<sourcetable>** is optional. For the JDBC or SAP Extractor, a single source table and at least one source column must be specified. Otherwise, an error message will be output during the parsing of the data source file.

A data source definition can only have a maximum of one table. It is impossible to restrict the number of rows. All rows are extracted, including all rows with identical values at the columns to be extracted. For example, if the columns **First name** and **Last name** are to be extracted and the table contains ten entries **Peter** and **Schmidt**, ten events with identical attribute values will be generated.

The following example explains the configuration:

```
<realmtable tablename="COMPANY_EMPLOYEE">
   <sourcetable tablename="EMPLOYEE">
      <sourcefield>EMPLOYEE_ID</sourcefield>
      <sourcefield>NAME</sourcefield>
   </sourcetable>
   ...
</realmtable>
<dataextraction>
   <outputfilename>..\custom\testclient\data\employee.xml</outputfilenam
   e>
</dataextraction>
...
<systemconfig>..\custom\testclient\SourceSystemConfig.xml</systemconfig>
```

In contrast to the behavior of the conventional JDCB or SAP Extractor, attributes no longer receive a table name as a prefix in the event output file. For example, if the table **EMPLOYEE** was extracted, the extractor usually generated events of the type **<table name>-<column name>**:

```
<event>
   <attribute type="EMPLOYEE-EMPLOYEE_ID">4711</attribute>
   <attribute type="EMPLOYEE-NAME">Schmidt</attribute>
</event>
```

Extracting a table using the **<realmtable>** element, however, creates only events without table names:

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
```

NULL VALUES IN THE DATA ANALYTICS MODE

If the value of a column is **null** at the time an analysis realm table is extracted, this value is not written to the event. If a row **EMPLOYEE_ID = 4712** exists without last name, the extractor creates the following events.

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
<event>
   <attribute type="EMPLOYEE_ID">4712</attribute>
</event>
```

However, if all column values to be extracted are **null**, the event is written with empty attributes:

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
<event>
   <attribute type="EMPLOYEE_ID"></attribute>
   <attribute type="NAME"></attribute>
</event>
```

If multiple such rows exist they will be – in contrast to the common extraction procedure (analysistype=PROCESS) – transferred accordingly so that as many <event> elements exist in the event file as there are rows in the data table of the source system.

4.3.15.1 Restrict data extraction

If you want to extract only a specific part of the table content, you can define data extraction conditions. You can limit the data volume to be extracted using a time stamp or integer value (for example, a sequence).

To do so, specify the required conditions in the data source configuration using the **condition** element and the **dataextractiontype** attribute.

```
<datasource name="VBAP" type="JDBC" analysistype="DATA_ANALYTICS"
dataextractiontype="TIME_BASED">
      <realmtable tablename="VBAP">
            <sourcetable tablename="VBAP2">
                  <condition logicaloperator="char_creationtimestamp"
fieldname="AEDAT">
                        <value>yyyy-MM-dd</value>
```

```
                </condition>
                <sourcefield>AEDAT</sourcefield>
                <sourcefield>ERDAT</sourcefield>
                <sourcefield>ERNAM</sourcefield>
                <sourcefield>ERZET</sourcefield>
                <sourcefield>MATKL</sourcefield>
                <sourcefield>MATNR</sourcefield>
                <sourcefield>WERKS</sourcefield>
            </sourcetable>
        </realmtable>
...
</datasource>
```

The **dataextractiontype** attribute can have the following values in Data analytics data sources:

- COMPLETE: Extraction is executed with a time criterion from the date **01.01.1990 00:00:00** and with an integer criterion from the value **0**. This is also the default setting if the attribute does not exist.
- TIME_BASED: A data extraction based on a time stamp is to be run.
- VALUE_BASED: A data extraction based on a value is to be run.

There is no check whether the data extraction condition used matches the data extraction type configured (dataextractiontype=). If the configuration is created using CTK, CTK ensures correct values. If the file is configured manually and a condition that does not match is used, an error may occur during data extraction.

4.3.16 Extract tables with key/value columns

You can use a special extraction class to extract tables that contain cells with keys/value pair. A new event attribute is created for each extracted key of an event.

The following example shows a table in the source system that contains key value attributes for an event.

Table "WF_ITEMS" (system event table, header table for work items)

ID (key)	TYPE	TEXT	CREATOR	STAT	…
000000532657	F	Create a benefit adjustment reason for employee 00001448 due to any event.	Creating an adjustment reason due to event...	COMPLETED	…
…	…	…	…	…	…

For each of these work items (that is, for each line of the "WF_ITEMS" table), there are arbitrary attributes that are stored in the table "WF_ITEM_ATTRIBUTES" as key/value:

Table "WF_ITEM_ATTRIBUTES"

ID (key)	ATTRIBUTE (key)	VALUE	...
000000532657	DATEOFVALIDITY	20020101	...
000000532657	EMPLOYEENUMBER	00001448	...
000000532657	EVENT	CREATED	...
000000532657	OBJECTTYPE	FAMILY	...
000000532657	_WF_INITIATOR	USWF-BATCH	...
000000532657	_WF_PRIORITY	5	...
...	...	...	...

EXTRACTION CLASS

com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableKeyValueFieldsToAttributes_sap2ppm

This extraction class creates a new attribute for the source system event from each key value found, and the value becomes the value of the attribute. For this class, you must specify the names of the key and value column as parameters separated by "#-#".

All key and value fields must be specified as fields to be read (XML element **fieldtoread**). Additional fields to be read are not allowed.

The following configuration refers to the JDBC extraction class mentioned above:

```
<xmlextractor_tableconfiguration>
    <configuration name="workflow">
        <docspec>
            <doctable name="WF_ITEMS" tablename="WF_ITEMS">
                <pkfield name="ID" />
                <pkfield name="TYPE" />
                <pkfield name="CREATOR" />
                <pkfield name="TEXT" />
            </doctable>
        </docspec>
        <table name="WF_ITEM_ATTRIBUTES" tablename="WF_ITEM_ATTRIBUTES" classtouse=
        "com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableKeyValueFieldsToAttributes_sap2ppm">
            <parameter name="KEYVALUEFIELDS">
                <value>ATTRIBUTE#-#VALUE</value>
            </parameter>
            <pkfield name="ID" fktablename="WF_ITEMS" fkfieldname="ID" />
            <fieldtoread name="ATTRIBUTE" />
            <fieldtoread name="VALUE" />
        </table>
    </configuration>
</xmlextractor_tableconfiguration>
```

The following example event was read out with the above configuration:

```
<event>
  <attribute type="WF_ITEMS-ID">000000532656</attribute>
  <attribute type="WF_ITEMS-CREATOR">Creating an adjustment reason due to
event...</attribute>
  <attribute type="WF_ITEMS-STAT">COMPLETED</attribute>
  <attribute type="WF_ITEMS-TEXT">Create a benefit adjustment reason for
employee 00001448 due to any event.</attribute>
  <attribute type="WF_ITEMS-TYPE">F</attribute>
'<attribute type="WF_ITEM_ATTRIBUTES-DATEOFVALIDITY ">20020101</attribute>
  <attribute type="WF_ITEM_ATTRIBUTES-EMPLOYEENUMBER
">00001448</attribute>
  <attribute type="WF_ITEM_ATTRIBUTES-EVENT ">CREATED</attribute>
  <attribute type="WF_ITEM_ATTRIBUTES-OBJECTTYPE ">FAMILY</attribute>
  <attribute type="WF_ITEM_ATTRIBUTES-_WF_INITIATOR
">USJOSWIGT</attribute>
  <attribute type="WF_ITEM_ATTRIBUTES-_WF_PRIORITY ">5</attribute>
</event>
```

SPECIAL CHARACTERS IN KEYS

The values of the key columns are used in the **type** XML attribute of the **attribute** XML elements. These values can contain characters that must not be used in this XML attribute. For this reason, the values of the **type** attribute of the **attribute** XML element are checked for invalid XML characters and such characters are filtered out if necessary.

4.4 Command line program

You create the XML output file(s) using the **runsap2ppm.bat** command line program.

Calling up the program without parameters or with **-h** or **-?** outputs the online help on the console, describing all available options.

4.4.1 Command line program arguments

4.4.1.1 General arguments

-VERSION

Outputs the version number of the PPM process extractor used. Other arguments specified are ignored.

-INFORMATION YES|NO

Specify whether information is to be output (yes) or not (no) during the import. The default is **yes**.

-WARNING YES|NO

Specify whether warning messages are to be output (yes) or not (no) during the import. The default is **yes**.

-ERROR YES|NO

This is where you specify whether error messages are to be output (yes) or not (no) during the import. The default is **yes**.

-PROTOCOLFILE <FILE NAME>

Specify the log file to which all messages are written during the import. If you specify a file, only critical error messages resulting in program abortion will be output on the screen.

-LANGUAGE <ISO CODE>

Specify the language in which the log information is to be output.

4.4.1.2 Source system-specific arguments

-DATASOURCE <FILE NAME> (OPTIONAL)

Specify the data source you want to use for the extraction. The XML file contains specifications of all files to be used for extraction and XML output. Make sure that you use this parameter instead of and not in combination with the **-datasourcelist**, **-systemconfig**, **-tableconfig**, **-calcconfig**, **-outfile**, or **-nozip** parameters.

-DATASOURCELIST <FILE NAME>

Multiple data sources can be extracted simultaneously by means of the **-datasourcelist** argument. This corresponds to multiple consecutive extraction using the **-datasource** argument. Only SAP data sources are extracted.

See chapter Extract multiple data sources (page 70).

-SYSTEMCONFIG <FILENAME> <CONFIGNAME>

This is where you specify the name of the XML file containing the system configuration(s). The content of the file is source system-specific and can contain several configurations. The name of the configuration to be used is specified in the second argument. If the XML file contains only one configuration, this is used automatically. In this case you do not need to specify a configuration name.

-TABLECONFIG <FILE NAME> <CONFIG NAME>

Specify the name of the XML file containing the table configuration(s). The content of the file is source system-specific and can contain several configurations. The name of the table configuration is specified in the second argument. If the XML file contains only one configuration, this is used automatically. In this case you do not need to specify a configuration name.

-CALCCONFIG <FILE NAME> (OPTIONAL)

Specify the name of the XML file with which you can change attributes of the XML output file or add attributes including attribute transformations.

-BEGINDATE <DD.MM.YYYY> (OPTIONAL)

Specify the date from which data is to be extracted from the source system. If you do not use this parameter, the value of the **lastreaddate** XML attribute from the specified system configuration or data source is used. The SAP format **yyyymmdd** is supported.

-BEGINTIME <HH:MM:SS> (OPTIONAL)

Specify the time from which data is to be extracted from the source system. If you do not use this parameter and **-begindate**, the value of the **lastreadtime** XML attribute from the specified system configuration or data source is used. If you do not use the parameter but you do use **-begindate**, the default value **000000** is set.

-ENDDATE <DD.MM.YYYY> (OPTIONAL)

Specify the date up to which data is to be extracted from the source system. If you do not use this parameter, the current date is used. The SAP format **yyyymmdd** is supported.

-ENDTIME <HH:MM:SS> (OPTIONAL)

Specify the time up to which data is to be extracted from the source system. If you do not use the parameter but use
-enddate, **235959** is set as the default value. If you use neither this parameter nor **-enddate**, the current time is used.

If PPM Process Extractor SAP-2-PPM is started without specifying a period of time, the start time is extracted from the specified configuration file. The current date and time are used as the end time.

-VALUECONSTRAINT <PARAMETER1> ... <PARAMETER4> (OPTIONAL)

Use this parameter to delimit the data volume to be extracted using an integer criterion. The "<", "<=", ">" and ">=" comparison operators are allowed. You can compare the corresponding field value with one or two (interval) values. If a last extracted value (**lastreadvalue**) has

already been saved in the system configuration or data source, you can also use this value for the comparison (see chapter **Continuous automated extraction** (page 68)).

Examples

```
-valueconstraint 25000020 "<="
```

Only data records are extracted whose integer value for the field specified in the event specification is **smaller than** or **equal to** the value **25000020**.

```
-valueconstraint 15000020 ">" 25000020 "<"
```

Interval: Only data records are extracted whose integer value for the field specified in the system event specification is **smaller than** the value **25000020** and **greater than** the value **15000020**.

-SAVE_VALUE_MINIMUM (OPTIONAL)

Use this parameter to specify that with **-valueconstraint**, the smallest value last read is saved as **lastreadvalue** in the configuration file used (system configuration or data source) when extracting with restrictions. Default value: Saving the largest value

-CPD <INT> (OPTIONAL)

This parameter is used to specify the number of system events to be extracted simultaneously (concurrently processed documents). The specified value gradually extracts the primary key fields from the table referenced using the **doctable** XML element in the table configuration, and stores them in the extractor's Java memory. Specifying a higher value results in improved extraction performance and more efficient memory usage.

If you do not specify the parameter, the recommended default value of 1000 is used.

-PING (OPTIONAL)

This parameter is used to test the connection to the specified source system. No data is extracted.

-RFC_BLOCKMODE {G|D|M} (OPTIONAL)

This parameter enables you to use block data extraction (see chapter **Block extraction of large data volumes** (page 37)). Specify the appropriate mode as an argument. Available block modes:

G (internal table of the function group)
M (ABAP memory)
D (database table)

-PROGRESS {YES|NO} (OPTIONAL)

You use this parameter to specify whether or not you want to output additional log messages about the progress of block data extraction. Default value: **yes**

-RFC_CALLRETRYS <NUMBER> (OPTIONAL)

You use this parameter to specify the number of times that queries to the R/3 system are repeated if an error occurs or the connection fails. When using block modes **G** and **M**, the specified number must be **0**. The parameter is effective globally, not just for block extraction. Default value: **0**

-RFC_DELIMITER <CHARACTER> (OPTIONAL)

You use this parameter to specify the character to be used to separate the extracted values. The character may not be contained in the actual extracted values themselves. Default value: **;**

-REMOVEEMPTY (OPTIONAL)

You use this parameter to remove extracted attributes with no values before the attribute transformation.

-REMOVEZERODATES (OPTIONAL)

Use this parameter to remove the values of SAP date fields, that is, SAP table fields of the **DATS** or **D** type containing only zeros. The values will be removed from the events right after data extraction.

Example

<attribute type="VBAP-ABDAT">00000000</attribute>

4.4.1.3 Output file-specific arguments

-OUTFILE <FILE NAME>

You use this parameter to specify the name of the XML output file. The file extension is added automatically. By default, the file is output as a ZIP file.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

You use this parameter to specify the encoding of the XML output file. Default value: UTF-8

-NOZIP (OPTIONAL)

You use this parameter to specify that the output file is to be output as an XML file rather than as a ZIP file.

-PIKIDATAMAPPING <FILE NAME> <PC NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the measure series in which the process instance-independent measure series are to be saved.

-DIMDATAMAPPING <FILE NAME> <DIM NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the dimension for which the extracted values are subsequently to be imported.

-SORTEVENTATTRIBUTES (OPTIONAL)

You use this parameter to sort the source system attributes in the system events in alphanumeric order using the attribute type. For large data volumes, this parameter can have a negative impact on the extraction speed.

4.4.1.4 Continuous automated extraction

You can extract data in full and automatically by eliminating the **-begindate**, **-begintime**, **-enddate**, **-endtime** parameters in the command line or by using **-valueconstraint** with no value for the first comparison operator. If the data range to be extracted is restricted using integer values, either the smallest or largest of the last values extracted is saved or updated in the configuration file used. If there are time restrictions on the data range to be extracted, the time of the last data extraction is saved or updated in the configuration file.

The following conditions must be met:

- A correct configuration for **creationtimestamp** and **valueconstraint** for the data to be extracted is specified in the system event specification (see chapter **Condition operators** (page 26))
- You always use the same command line call for continuous automated extraction, for example:

```
runsap2ppm -datasource datasource.xml -valueconstraint ">"
```

- When you use the **-valueconstraint** parameter, make sure that you either save the greatest of the last values extracted for all extraction operations (default), or the smallest of the last values extracted by specifying **-save_value_minimum**. The value of **lastreadvalue** in the configuration file is only updated if the greatest of the last values extracted is greater than the value currently saved when running the command line without the **-save_value_minimum** parameter, or if the smallest of the last values extracted is smaller than the value currently saved when using **-save_value_minimum**.

Example 1 (lastreadvalue with default value "0")

```
-valueconstraint ">="
```

Only those data records are extracted whose integer value for restricting the data range to be extracted is **greater than** or **equal to** the value saved as the last value extracted (**lastreadvalue**) in the system configuration or data source. If no value is saved, the default value **0** is used (in the example, all data records with integer criterion values **>=0** would be extracted). The greatest of the last values extracted (**300** in this example) is saved as **lastreadvalue** according to the default setting (see **-save_value_minimum** parameter).

Example 2 (lastreadvalue="40")

```
-valueconstraint ">=" 270 "<="
```

All data records are extracted that have an integer value of **>=40** and **<=270** for restricting the data range to be extracted. After the extraction, the value for **lastreadvalue** is updated, for example, to **270** if this is really the greatest of the last values extracted.

Example 3 (lastreaddate="19971231" lastreadtime="155959" lastreadvalue="270")

By calling up

```
runsap2ppm -datasource datasource.xml -valueconstraint ">"
```

again all data records are extracted whose integer value for delimiting the data range to be extracted is greater than **270** and whose time stamp is as old as or more recent than the start time of the data extraction (**31.12.1997 15:59:59**).

The start time of data extraction is entered as **lastreaddate**/**lastreadtime** in the configuration file.

If you eliminate the **-begindate**/**-begintime** parameters, you always extract the data records whose time stamps are more recent (greater) than the ones specified and last extracted using **lastreaddate**/**lastreadtime**. The current date and time are entered as **lastreaddate**/**lastreadtime** if you do not use the **-enddate** and **-endtime** parameters.

4.5 Extract multiple data sources

You can use a data source list to extract multiple data sources simultaneously. The data source list is specified in a separate configuration file.

Each new SAP data source created in CTK is automatically added to the end of the list of data sources of the current client. During extraction using the command line option **-datasourcelist <file name>** the data of these data sources is extracted consecutively as if data extraction was called multiple times using the option **-datasource <file name>**. The sequence of the data source extraction is specified in the configuration file. Data sources of a type other than MYSAP in the list are ignored during extraction.

The following applies to trouble shooting:

- If the extraction is called via a valid data source list that does not contain any matching data sources the extraction ends without outputting an error message.
- If the extraction is called using a data source list containing multiple matching data sources, and if an error occurs during the extraction of one of these data sources resulting in extraction cancelation, the procedure continues with the next data source. This means that cancelation of the extraction of one data source does not lead to cancelation of the entire procedure.
- If an error occurs in at least one data source, which would have led to an exit error status during the single extraction of this data source, the last exit error status will be returned by the overall data source extraction process.

4.6 SAP Secure Network Communication

SAP-2-PPM supports SAP Secure Network Communication (SNC).) SAP SNC is a proprietary protocol from SAP to safeguard RFC connections.

You can use CTK to configure the required settings for importing data from data sources with SNC. Further information can be found in the CTK help.

4.6.1 Prerequisites

To enable SAP-2-PPM to read SAP data sources with SNC, the following requirements have to be met.

- Configure the SAP application server for SNC, including setting up appropriate SNC users.
- Install appropriate SAP-certified encryption software that is compatible with GSS API v2. We recommend using SAP Common Cryptolib, with which PPM has been tested.

- Configure the security environment as specified by the manufacturer of the installed encryption software. By default, this involves exchanging security certificates with the SAP application server.
- Use SAP JCo 3.0.14 or higher. We recommend using the latest version of JCo. Define the environment variables **SNC_LIB** and **SECUDIR** in your operating system environment. The RFC library embedded in JCo requires these variables to find the SNC layer and the required credentials.

Detailed information about SAP Common Cryptolib and SAP JCo can be found in the corresponding SAP documentation.

4.6.2 XML configuration of SAP data sources

The definition of the system configuration (R3Config.dtd) for a SAP data source with SNC is extended with the following attributes:

```
<!ATTLIST r3system
…
snc_mode (true|false) "false"
      snc_sso (true|false) "true"
      snc_qop (1|2|3|8|9) "9"
      snc_partnername CDATA #IMPLIED
snc_myname CDATA #IMPLIED
…
>
```

Attribute	Description
snc_mode	Enable/disable SNC Default: Disabled
snc_sso	SNC logon with user name and password (without single sign-on) Default: Use SSO
snc_qop	SNC service quality 1: Authentication only 2: Integrity protection 3: Confidentiality protection 8: Default specified by application server 9: Highest available security setting (default)
snc_partnername	SNC name of the application server (for example, p:CN=R3, O=XYZ-INC, C=US)
snc_myname	SNC name of the RFC client (for example, p:CN=Smith, O=XYZ-INC, C=EN)

The **user** attribute previously required is now optional. Specifying the user name and password is not required for SNC unless it is specifically set using the **snc_sso=false** attribute.

The SNC configuration for PPM-2-SAP extractor data sources and for SAP data sources for PPM Data Analytics is identical.

Here is an example of a complete configuration when using SNC:

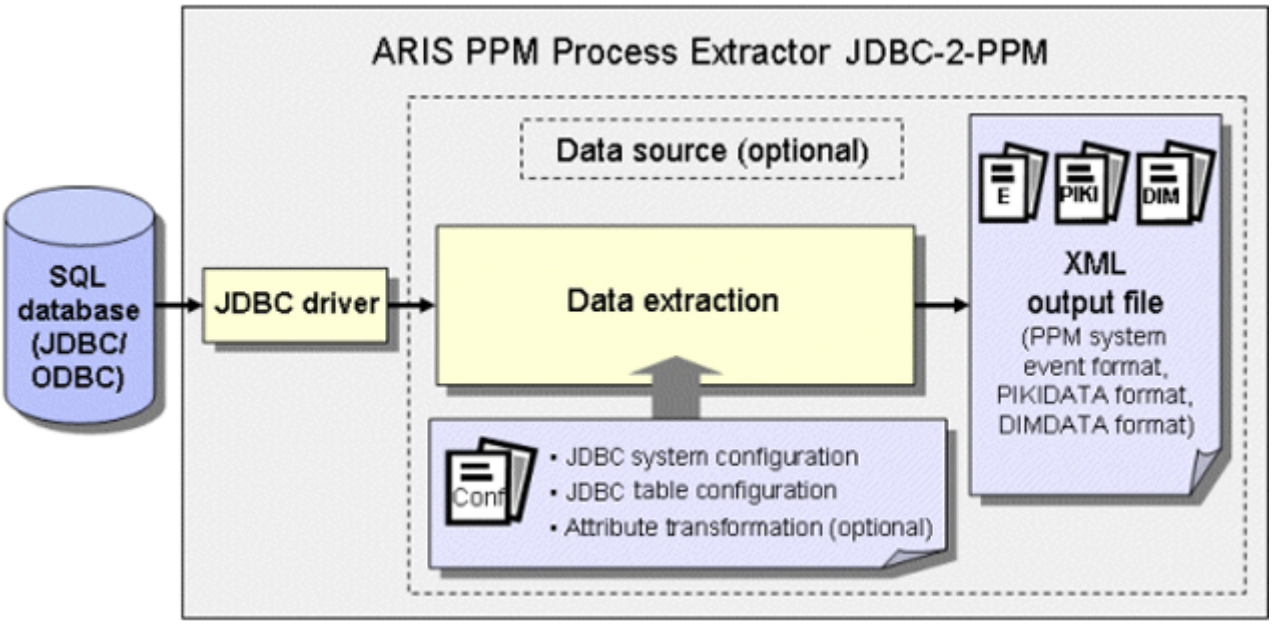
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE r3systemconf SYSTEM "R3Config.dtd">
<r3systemconf>
  <r3system configname="snc_connection"
  appserver="xyz.com" systemnumber="00"
  client="777" user="" snc_mode="true" snc_sso="true" snc_qop="9"
  snc_partnername="p:CN=R3, O=XYZ-INC, C=US"
  snc_myname="p:CN=Smith, O=XYZ-INC, C=DE"
  language="" lastreaddate="19700101" lastreadtime="000000"
  encryptpw="" />
</r3systemconf>
```

5 PPM Process Extractor JDBC-2-PPM

This chapter provides an overview of the architecture, functioning and configuration of PPM Process Extractor JDBC-2-PPM.

5.1 Architecture

The figure below illustrates the functionality of PPM Process Extractor JDBC-2-PPM:



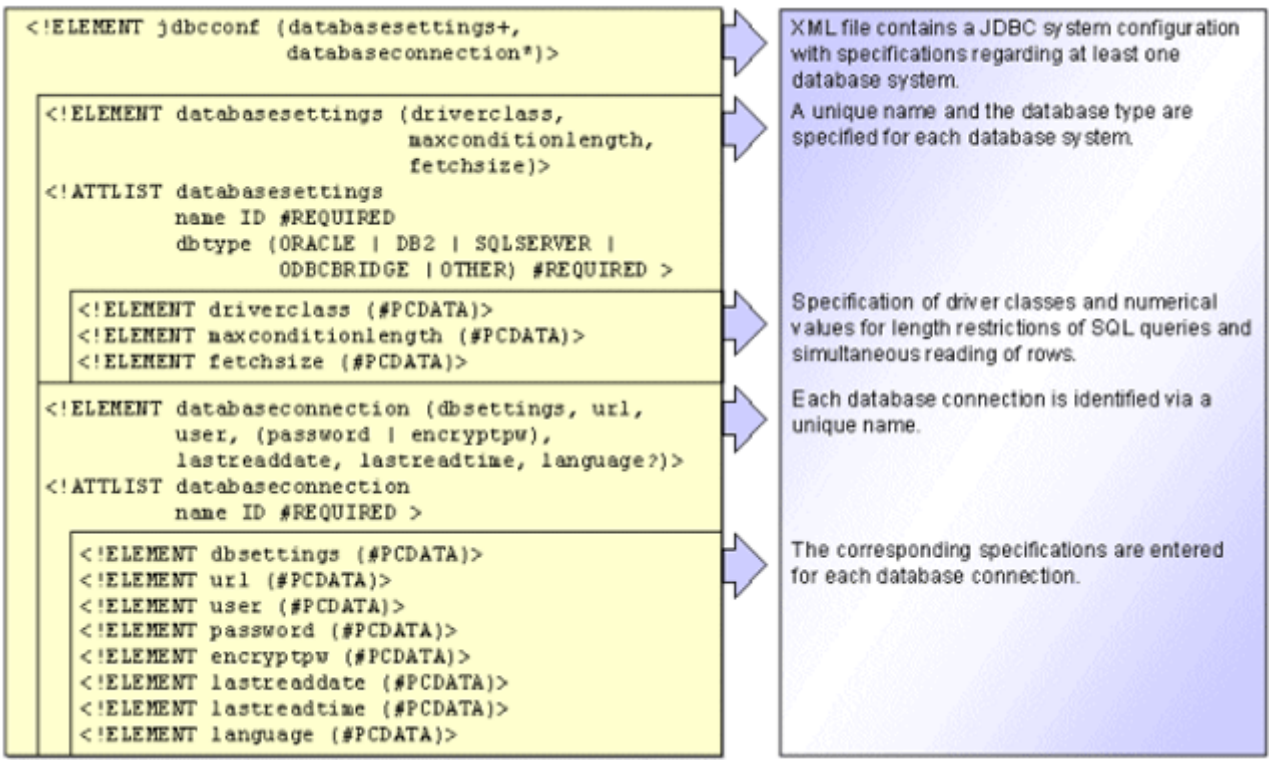
The JDBC driver is used to create a connection to the SQL source database with the access data for the JDBC system configuration. Data is then extracted from the tables in the source database system using the JDBC table configuration and an optional transformation using the **data extraction** (page 4) module and is written to an XML file in a PPM-compatible output format.

You need to ensure that the specified source system user has appropriate access authorization to extract the relevant data fields.

5.2 JDBC system configuration

The access data for the SQL database system is specified in an XML file. The name of this XML file is transferred to the command line tool as an argument.

The format of the XML file is specified by the following DTD:



XML element or attribute	Description	Example
jdbcconf	JDBC system configuration specifying the database systems and database connections to be used	-
databasesettings	General settings for the database system used	-
name	Unique name of the general database system settings	setting_oracle
dbtype	Database type used; valid values: **ORACLE** | **DB2** | **SQLSERVER** | **OTHER**	ORACLE
driverclass	Driver class to be used for the database system	oracle.jdbc.driverOracleDriver
maxconditionlength	The specified figure limits the length of the WHERE component of an SQL query	1000

XML element or attribute	Description	Example
fetchsize	Number of data records transferred simultaneously from the DB server to the Java client. For most database systems, the default value is **10**. Recommended value: **1000**	1000
databaseconnection	Information on the database connection	-
name	Unique name of the database connection	oracle_connection1
dbsettings	Unique name of the general database system settings	setting_oracle
url	URL of the database connection	See file extract below
user	Name of the database user	ppmoracle
password	Password of the database user	ppmoracle
encryptpw	Encrypted password as a combination of figures after evaluation of the <password> element	62 -62 -56
lastreaddate	Date from which data is extracted. Not applicable when using a **data source** (page 158), format: **yyyymmdd**	20050131
lastreadtime	Time from which data is extracted. Not applicable when using a **data source** (page 158), format: **hhmmss**	152917
language	Language of text fields extracted	EN

The database type can be one of the preset values for **Oracle**, **IBM DB2**, or **MS SQL Server** database systems. The **OTHER** value should be used if you want to extract data from a database of a type other than those listed above.

The **maxconditionlength** parameter enables you to specify a figure that limits the length of the WHERE component of an SQL query to the specified value. This is necessary to comply with any length restrictions in the different database systems. If the SQL query is longer than the specified maximum length, it is split into several smaller queries that are executed separately.

The **fetchsize** parameter enables you to specify how many lines of data will be simultaneously transferred from the database to the JDBC extractor per read operation.

The password of the database user must be specified in unencrypted form and plain text in the **password** XML element. After evaluation during the following extraction, the password is written back to the JDBC system configuration in encrypted form in the **encryptpw** XML element. The **password** entry is then deleted. At any time, the system configuration may only include one of the two entries.

To change an existing password, add a new **password** entry to the current configuration file and delete the **encryptpw** entry. The new password is encrypted as part of the next extraction.

The **language** parameter is optional. If you specify a value for it, this is compared with the field name from the table configuration (**langfieldname**) (see chapter **JDBC table configuration** (page 77)). For example, if **langfieldname="LANG"** is specified in a table and **<language>EN</language>** is specified in the JDBC system configuration, only those data records for which **LANG** has the value **EN** will be extracted.

The following file extract shows an example JDBC system configuration:

```
<jdbcconf>
  <databasesettings name="setting_sqls" dbtype="SQLSERVER">
    <driverclass>com.microsoft.jdbc.sqlserver.SQLServerDriver
    </driverclass>
    <maxconditionlength>1000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databasesettings name="setting_oracle" dbtype="ORACLE">
    <driverclass>oracle.jdbc.driver.OracleDriver</driverclass>
    <maxconditionlength>1000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databaseconnection name="sqls_connection">
    <dbsettings>setting_sqls</dbsettings>
    <url>jdbc:microsoft:sqlserver:
      //PC3:1433;SelectMethod=Cursor;
                DatabaseName=ppmdb
    </url>
    <user>ppmuser</user>
    <password>ppmuser</password>
    <lastreaddate>20050228</lastreaddate>
    <lastreadtime>000000</lastreadtime>
    <language />
  </databaseconnection>
```

```
  <databaseconnection name="oracle_connection1">
    <dbsettings>setting_oracle</dbsettings>
    <url>jdbc:oracle:thin:@pcppm:1521:orappm_test
    </url>
    <user>ppmoracle</user>
    <password>ppmoracle</password>
    <lastreaddate>20051231</lastreaddate>
    <lastreadtime>235959</lastreadtime>
  </databaseconnection>
  <databaseconnection name="oracle_connection2">
    <dbsettings>setting_oracle</dbsettings>
    <url>jdbc:oracle:thin:@pcppm:1521:orappm_produktiv</url>
    <user>ppmoracle2</user>
    <encryptpw>48 -62 -76 -60 -108 -57 -92</encryptpw>
    <lastreaddate>20050228</lastreaddate>
    <lastreadtime>000000</lastreadtime>
  </databaseconnection>
</jdbcconf>
```

For PPM Process Extractor JDBC-2-PPM to be able to use the configuration data to connect to the database, you need to copy the JDBC drivers (JAR and/or ZIP files) to the following directory of your installation.

<PPM installation directory>\ppm\server\bin\work\data_ppm\drivers

5.3 JDBC table configuration

The JDBC table configuration (system event specification) specifies which table fields are extracted from the SQL database system and written to the system events as source system attributes. You can save several table configurations with unique names in the XML file.

The JDBC table configuration consists of the following components:

GLOBAL TABLES

Global tables are used to extract information that is written for all system events.

FOREIGN KEY TABLES

The **docreftable** XML element contains the name of the foreign key table. It specifies how the data area to be extracted from the system event table is limited. The primary key fields specified in the **pkfield** XML element link the foreign key table to the system event table and other foreign key tables.

SYSTEM EVENT TABLE

The **doctable** XML element contains the name of the system event table. It specifies the documents to be extracted for a document flow. Each data record extracted from the system event table generates a system event in the output file (**event** XML element).

DATA TABLES

The information in the system event table can be supplemented by extracting additional data fields from any other data tables (for example, the material number is extracted from the system event table and the descriptive text relating to this number is extracted from a data table).

Some database systems allow table names that do not comply with the SQL standard. Extracting such tables with PPM Process Extractor JDBC-2-PPM results in an error message. You can extract contents by creating a view for each table to be extracted that contains names that are not SQL-compliant, and then use the view for extracting with PPM Process Extractor JDBC-2-PPM.

The following XML file framework illustrates the configuration of the tables from which data is to be extracted. For details about which XML elements or attributes are optional, please refer to the explanatory table in chapter **Table access configuration** (page 80).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE r3systemconffields SYSTEM
          'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="..." printname="..." classtouse="...">
    <globaltable name="..." tablename="..." classtouse="...">
      <fieldtoread name="..."/>
         <textref tablename="..." reffieldname="..."
                  textfieldname="..." langfieldname="..."/>
      </fieldtoread>
      ...
    </globaltable>
    <docspec>
      <docreftable name="..." tablename="..."
                                    classtouse="...">
        <condition fieldname="..." logicaloperator="...">
          <value>...</value>
        </condition>
        <pkfield name="..." fktablename="..."
                                   fkfieldname="...">
          <fkpart readfrom="..."
                      startposition="..." length="..."/>
          <prefix>
            <value>...</value>
          </prefix>
          <postfix>
            <value>...</value>
          </postfix>
        </pkfield>
        ...
```

```
      </docreftable>
      ...
      <doctable name="..." tablename="..."
                                      classtouse="...">
        <condition fieldname="..." logicaloperator="...">
          <value>...</value>
        </condition>
        <pkfield name="..." fktablename="..."
                                    fkfieldname="...">
          <fkpart readfrom="..." startposition="..."
                                       length="..."/>
          <prefix>
            <value>...</value>
          </prefix>
          <postfix>
            <value>...</value>
          </postfix>
        </pkfield>
        ...
        <fieldtoread name="...">
          <textref tablename="..." reffieldname="..."
                   textfieldname="..." langfieldname="..."/>
        </fieldtoread>
        ...
      </doctable>
    </docspec>
    <table name="..." tablename="..." classtouse="...">
      <condition fieldname="..." logicaloperator="...">
         <value>...</value>
      </condition>
      <pkfield name="..." fktablename="..."
                                    fkfieldname="...">
         <fkpart readfrom="..." startposition="..."
                                       length="..."/>
         <prefix>
           <value>...</value>
         </prefix>
         <postfix>
           <value>...</value>
         </postfix>
      </pkfield>
      ...
      <fieldtoread name="...">
        <textref tablename="..." reffieldname="..."
                  textfieldname="..." langfieldname="..."/>
      </fieldtoread>
      ...
    </table>
    ...
  </configuration>
  ...
</jdbc_tableconfiguration>
```

5.3.1 Table access configuration

XML element	XML attribute	Description
jdbc_table-configuration		List of table configurations
configuration	name	Unique identifier of the table configuration
	printname (optional)	Switch. **yes** means that the configuration name is output as a prefix to the field names. Default value: **no**
	classtouse (optional)	Name of the Java class that is used for editing the configuration Default value: Standard implementation
globaltable (optional)	name	Identifier of the table from which global system attributes are extracted
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
docspec	-	Grouping of the specifications of all foreign key tables and the system event table
docreftable	name	Identifier of a foreign key table
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute

XML element	XML attribute	Description
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
doctable	name	Identifier of the system event table. Each line extracted creates a system event in the XML file.
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
table (optional)	name	Identifier of the data table from which supplementary information is extracted
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation

Each table may only be referenced once in the configuration. In order to be able to extract a table multiple times, reference the same table (**tablename** XML attribute) using different identifiers.

Example

You want to extract the **VBAP-MATNR_1** and **VBAP-MATNR_2** fields from the **PRODH** table. The file extract below illustrates the configuration:

```
<table name="PROD_HIER_1" tablename="PRODH"
                              classtouse="ZTable">
   <pkfield name="HNUM" fktablename="VBAP"
                           fkfieldname="MATNR_1"/>
   <fieldtoread name="HVAL"/>
</table>

<table name="PROD_HIER_2" tablename="PRODH"
                              classtouse="ZTable">
   <pkfield name="HNUM" fktablename="VBAP"
                           fkfieldname="MATNR_2"/>
   <fieldtoread name="HVAL"/>
</table>
```

5.3.2 Global meta data

The **globaltable** XML element can be used to transfer meta data from the PPM SAP-2-PPM and JDBC-2-PPM process extractors to the output file as global system event attributes. Typical data for global system event attributes include the names of the configurations that were used to extract the data. The data itself is determined from the argument values specified in the command line. For arguments that are not specified in the command line, the default value is determined.

The identification of keywords is case-sensitive.

Global system event attributes apply to all system events in the current output file.

Keyword	Value from command line argument
SYSTEM_CONFIG_FILE_NAME	-systemconfig <filename> ...
SYSTEM_CONFIG_NAME	-systemconfig ... <configname>
TABLE_CONFIG_FILE_NAME	-tableconfig <filename> ...
TABLE_CONFIG_NAME	-tableconfig ... <configname>
BEGIN_DATE,	-begindate <dd.MM.yyyy>
BEGIN_TIME	-begintime <hh:mm:ss>
END_DATE	-enddate <dd.MM.yyyy>
END_TIME	-endtime <hh:mm:ss>
CPD	-cpd <int>
OUT_FILE_NAME	-outfile <file name>
PIKI_DATA_MAPPING_FILE_NAME	-pikidatamapping <filename>...

Keyword	Value from command line argument
PIKI_DATA_MAPPING_NAME	-pikidatamapping ... <pcname>
DIM_DATA_MAPPING_FILE_NAME	-dimdatamapping <filename> ...
DIM_DATA_MAPPING_NAME	-dimdatamapping ... <dimname>

If no value can be determined for a keyword, the keyword itself will be written to the output file. In this way you can transfer constants to the output file.

Example

The file extract below shows the configuration that is used to create global system event attributes.

```
<globaltable name="INFO">
  <fieldtoread name="SYSTEM_CONFIG_NAME"/>
</globaltable>
```

The configuration name specified in the command line using -systemconfig **<configname>**, for example, **sapppm** is written to the output file as the **INFO-SYSTEM_CONFIG_NAME** global system event attribute:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <attribute type="INFO-SYSTEM_CONFIG_NAME ">
    sapppm
  </attribute>
  ...
  <event>
    ...
  </event>
  ...
</eventlist>
```

By specifying a suitable Java class in the **classtouse** XML attribute, data can also be extracted from any other source system table.

5.3.3 Sort system events

You can sort the system events in the XML output in ascending alphanumeric order based on the content of fields you define as sorting criteria. It is possible to specify multiple fields as sorting criteria. When sorting, the fields are prioritized from the first specified to the last specified, that is, the data is first sorted based on the content of the first field specified, then by the content of the second field specified and so on.

Tip

You can use the sorting option to create a more efficient XML import of the extracted data into the PPM system. The fact that system events belonging to a process instance follow each other immediately due to sorting in the XML output files means that they can be imported directly into an EPC using the process key during the XML import.

In the table configuration, in the **parameter** XML element, fields are specified as sorting criteria for the system event table (**doctable**) and the referenced foreign key tables (**docreftable**).

Example (extract from table configuration)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="BANF">
    <docspec>
      <doctable name="EBAN">
        <parameter name="ORDER_BY">
          <value>BNFPO</value>
          <value>BANFN</value>
        </parameter>
        <condition fieldname="ERDAT"
                   logicaloperator="creationtimestamp">
          <value>yyyyMMdd</value>
        </condition>
        <pkfield name="BANFN"/>
        <pkfield name="BNFPO"/>
        <pkfield name="ERDAT"/>
        <fieldtoread name="BSART"/>
        <fieldtoread name="KONNR"/>
        <fieldtoread name="KTPNR"/>
        <fieldtoread name="LIFNR"/>
        <fieldtoread name="MFRNR"/>
      </doctable>
    </docspec>
  </configuration>
</xmlextractor_tableconfiguration>
```

As specified in the **BANF** configuration (**parameter** XML element), the system events are written to the output file(s) sorted in ascending order by the item number in the purchase requisition number (**BNFPO** field) and, as a second priority, by the purchase requisition number (**BANFN** field).

The table below shows all of the configuration options:

XML element/attribute	Description
parameter	Specifies a sorting parameter for a system event table (**doctable**) or a foreign key table (**docreftable**).

XML element/attribute	Description
name	Parameter name. It is essential to specify **ORDER_BY**.
value	Specifies a sorting field. The field must be specified as a primary key field (**pkfield**).

You can set the sorting parameter in PPM Customizing Toolkit. To do this, activate the **System event** tab in the **Data extraction** module of the **Process merge** module group. Switch to editing mode and select **Edit system event** from the pop-up menu of the system event table. In the **Specify parameter** step, you can specify your settings.

5.3.4 Concatenate new attribute values from extracted attribute values

The **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableConcatFKs_jdbc2ppm** class can be used when extracting data from an JDBC system to generate new system event attributes by concatenating the values of attributes already extracted. The new system events created can be referenced in other data table configurations (**table** XML element).

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.ppm.
                        xmlextractortools.extractor.
                    jdbc2ppm.ZTableConcatFKs_jdbc2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
```

```
  </configuration>
</xmlextractor_tableconfiguration>
```

If you are using the **ZTableConcatFKs_jdbc2ppm** class, you must specify exactly one **fieldtoread** element. The element contains the name of the system event attribute to be created. In the example, a new system event attribute with the name **RBKP_NEW-AWKEY_GENERATED** is created. The value of the attribute consists of the values of the **pkfield** elements specified.
The normal **fkpart**, **prefix** and **postfix** operations can be used for the **pkfield** elements.

A system event created with the above configuration could look like this:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED">51056012042005
  </attribute>
</event>
...
```

The new attribute – **RBKP_NEWER-AWKEY_GENERATED** – can then be referenced in other **table** configurations in the same way as every other JDBC table field.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.
                ppm.xmlextractortools.extractor.
                   jdbc2ppm.ZTableConcatFKs_jdbc2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
    <table name="BKPF">
      <pkfield name="AWKEY" fktablename="RBKP_NEW"
                     fkfieldname="AWKEY_GENERATED"/>
      <fieldtoread name="BELNR"/>
    </table>
  </configuration>
```

```
</xmlextractor_tableconfiguration>
```

A system event created with this configuration could look like this:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED"
                      >5105601204 2005</attribute>
  <attribute type="BKPF-BELNR">5100004691</attribute>
</event>
...
```

5.3.5 Extraction using conditions

Extracting data from the tables can be limited by the use of conditions. For example, you can use conditions to control that, when extracting table fields, only document flows of one document type in a particular period are extracted.

The conditions are specified directly in the table configuration in the **booleancondition** or **condition** XML element and only apply to the associated table.

A condition contains the name of the table field, the comparison operator, and a concrete value. Conditions can be linked with any degree of complexity (**booleancondition** XML element).

XML element	XML attribute	Description
booleancondition (optional)	logicaloperator	Logical operators: **AND**, **OR**, **NOT** Default value: **AND**

XML element	XML attribute	Description
condition (optional)	logicaloperator	Comparison operators: **eq**, **neq**, **in**, **notin**, **num_gt**, **num_geq**, **num_lt**, **num_leq**, **is_null**, **is_not_null**, **timestamp_eq**, **timestamp_geq**, **timestamp_gt**, **timestamp_leq**, **timestamp_lt**, **time_eq**, **time_geq**, **time_gt**, **time_leq**, **time_lt**, **date_eq**, **date_geq**, **date_gt**, **date_leq**, **date_lt**, **num_eq**, **num_neq**, **num_in**, **num_notin** Operators for restricting the data range to be extracted: **char_creationtimestamp**, **date_creationtimestamp**, **valueconstraint** Default value: **eq**
	fieldname	Table field name

The **docbooleancondition** and **doccondition** XML elements enable conditional extraction from a data table depending on table fields already extracted. They are configured in a similar way to **booleancondition** and **condition**. The table name and the name of the column containing the data fields already extracted are specified in the **tablename** and **fieldname** XML attributes.

XML element	XML attribute	Description
docbooleancondition (optional)	logicaloperator	Logical operators: **AND**, **OR**, **NOT** Default value: **AND**
doccondition (optional)	logicaloperator	Comparison operators: **eq**, **neq**, **in**, **notin**, **exists**, **notexists**, **is_null**, **is_not_null**, Default value: **eq**
	tablename	Name of the table containing the table fields already extracted
	fieldname	Column name

5.3.5.1 Condition operators

Common operators

The following comparison operators are supported by the **condition** and **doccondition** XML elements:

Operator	Description
eq	Field content is equal to the specified value.
neq	Field content is not equal to the specified value.
in	Field content is equal to a specified value from a set of values.
notin	Field content is not equal to a specified value from a set of values.
is_null	Checks whether the field content is NULL
is_not_null	Checks whether the field content is not equal to NULL

Condition operators

Operator	Description
num_gt	Field content is greater than specified value.
num_geq	Field content is greater than or equal to the specified value.
num_lt	Field content is less than specified value.
num_leq	Field content is less than or equal to the specified value.
num_eq	Field content is equal to the specified value.
num_neq	Field content is not equal to the specified value.
num_in	Field content is equal to a specified value from a set of values.
num_notin	Field content is not equal to a specified value from a set of values.

Operator	Description
timestamp_eq	Checks whether the time stamp of the field corresponds to the specified comparison value (**value**). The comparison value must be specified in **dd.MM.yyyy HH:mm:ss** format – this also applies to all other timestamp_* operators.
timestamp_geq	Checks whether the time stamp of the field is greater than or equal to the comparison value.
timestamp_gt	Checks whether the time stamp of the field is greater than the comparison value.
timestamp_leq	Checks whether the time stamp of the field is less than or equal to the comparison value.
timestamp_lt	Checks whether the time stamp of the field is less than the comparison value.
time_eq	Checks the field's time for equality. The comparison value must be specified in **HH:mm:ss** format – this also applies to all other time_* operators.
time_geq	Checks whether the time of the field is greater than or equal to the comparison value
time_gt	Checks whether the time of the field is greater than the comparison value
time_leq	Checks whether the time of the field is less than or equal to the comparison value
time_lt	Checks whether the time of the field is less than the comparison value
date_eq	Checks whether the date of the field is equal to the comparison value. The comparison value must be specified in **dd.MM.yyyy** format – this also applies to all other date_* operators.
date_geq	Checks whether the date of the field is greater than or equal to the comparison value
date_gt	Checks whether the date of the field is greater than the comparison value

Operator	Description
date_leq	Checks whether the date of the field is less than or equal to the comparison value
date_lt	Checks whether the date of the field is less than the comparison value
like (only with string database fields, for example, CHAR, VARCHAR)	Comparison of field values with a variable string The following placeholders are permitted: * No or any number of characters ? Any single character \ Masking character for searching for placeholders or masking characters in the form: \\ or * or \? Example: `<condition fieldname="OBJECTID" logicaloperator="like"><value>*10?0\\20?0*</value></condition>` The search is performed for values such as 5551050\20106667 or 1080\204044 but not 34510550\2030*
char_ creationtimestamp	Extracts time stamps (date and time) from source database fields of the CHAR/VARCHAR type. Their values form the basis for restricting the data range to be extracted with the **-begindate** (**-begintime**) or **-enddate** (**-endtime**) command line parameters [see chapter **Source database specific arguments** (page 122)]. Multiple fields are separated by the character combination **#-#**. For each field, the time stamp format is specified in the **value** XML element. This format contains the values in the corresponding source database field. Example: `<condition fieldname="VC_DATE#-#VC_TIME" logicaloperator="char_creationtimestamp"><value>yyyyMMdd</value><value>HHmmss</value></condition>` When using the **char_creationtimestamp**

Operator	Description
	operator, be aware of possible effects of certain source database field time formats on data sorting and extraction. The database fields in a time format are extracted in alphabetical order. For example, if the date field values in the database are saved in **ddMMyyyy** format, the date **23021999** would be extracted when extracting the time period between **15.01.2000 - 31.12.2000** because in alphabetical terms, this date lies between the start and end date. However, the date **09122000** is not in this range because it is interpreted as earlier than the start date.
date_ creationtimestamp	Time stamps (date and time) are extracted from source database fields with times that are of a database system-dependent time data type. Their values form the basis for restricting the data range to be extracted with the **-begindate** (**-begintime**) or **-enddate** (**-endtime**) command line parameters [see chapter **Source database specific arguments** (page 122)]. Multiple fields are separated by the character combination **#-#**. For each field to be extracted, the **value** XML elements are used to specify the one of the three time data types corresponding to the source database fields. Only the following data types in the specified combinations are permitted: **DATE**/**TIME** or **TIME**/**DATE** or **TIMESTAMP** or **DATE** Example: `<condition fieldname="CHG_DATE#-#CHG_TIME" logicaloperator="date_creationtimestamp"> <value>DATE</value> <value>TIME</value> </condition>`
valueconstraint	Specifies a source database system field with integer values that are used for delimiting the

Operator	Description
	data range to be extracted with the **-valueconstraint** command line parameter [see chapter **Source database specific arguments** (page 122)]. The field to be extracted must be of an integer data type. Example: `<condition fieldname="INTEGERFIELD" logicaloperator="valueconstraint"/>`

When using the **timestamp_*** , **time_*** and **date_*** operators from the above table, you need to know which data type in which format the value extracted from the corresponding database system (Oracle, IBM DB2, MS SQL Server) is based on.

The following tables provide an overview of the different DB data types and DB formats and show some examples of formats generated in the XML output file.

Oracle/TIMESTAMP data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 12:25:11	Time = Creation time of database field value
04:02:36	21.12.2005 04:02:36	Date = Creation date of database field value

Oracle/DATE data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 12:25:11	Time = Creation time of database field value

Value format written to database (example)	Resulting format in XML output file (example)	Note
04:02:36	21.12.2005 04:02:36	Date = Creation date of database field value

The Oracle **DATE** data type saves time stamp values exclusively in **dd.MM.yyyy HH:mm:ss** format. To extract the required data from an Oracle database, you need to configure the extract conditions accordingly when using **date_*** operators.

Example

You want to extract all data records with the date **16.09.2004**, regardless of the time. If you specify the following condition in the table configuration:

```
<condition fieldname="TACT_TDATE" logicaloperator="date_eq">
  <value>16.09.2004</value>
</condition>
```

only data records with the **16.09.2004 00:00:00** time stamp are extracted. To ensure that all records of the specified date are extracted, you need to re-write the condition as follows:

```
<booleancondition logicaloperator="AND">
  <condition fieldname="TACT_TDATE"
                    logicaloperator="date_geq">
  <value>16.09.2004</value>
  </condition>
  <condition fieldname="TACT_TDATE"
                     logicaloperator="date_lt">
  <value>17.09.2004</value>
  </condition>
</booleancondition>
```

With the specified condition, all data records with the date **16.09.2004** and any time are extracted.

IBM DB2/TIMESTAMP data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005 04:02:36	07.05.2005 04:02:36	-

IBM DB2/DATE data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005	07.05.2005	-

IBM DB2/TIME data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
04:02:36	04:02:36	-

MS SQL Server/DATETIME data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 00:00:00	Time is always 00:00:00
04:02:36	01.01.1900 04:02:36	Date is always 01:01:1900

MS SQL Server/SMALLDATETIME data type

Value format written to database (example)	Resulting format in XML output file (example)	Note
07.05.2005 04:02:36	07.05.2005 04:02:00	Time only correct to the minute
07.05.2005	07.05.2005 00:00:00	Time is always 00:00:00
04:02:36	01.01.1900 04:02:00	Date is always 01.01.1900 and time is correct to the minute

doccondition operators

Operator	Description
exists	Field exists.
notexists	Field does not exist.

5.3.5.2 Complex conditions

The file extract below illustrates nested conditions using the example of the foreign key table:

```
...
  <docreftable tablename="VBAK" classtouse="ZDocRefTable">
    <booleancondition logicaloperator="AND">
      <condition fieldname="ERDAT#-#ERZET"
              logicaloperator="char_creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="VBTYP" logicaloperator="in">
        <value>C</value>
        <value>K</value>
      </condition>
      <booleancondition logicaloperator="OR">
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>1000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="eq">
            <value>0041</value>
          </condition>
        </booleancondition >
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>2000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="neq">
            <value>0060</value>
          </condition>
        </booleancondition >
      </booleancondition >
    </booleancondition >
    ...
  </docreftable>
...
```

In terms of propositional logic, the file extract shown corresponds to the following condition:

ERDAT and ERZET contain the creation time

and

VBTYP is C or V

and

((VKORG is equal to 1000 **and** VKBUR is equal to 0041) **or** (VKORG is equal to 2000 **and** VKBUR is not equal to 0060)).

5.3.6 Extract database fields

The database fields in the system event table and the linked data tables, from which values are to be extracted, are specified in the **fieldtoread** XML elements. For each **fieldtoread** XML element, a line of the form

```
<attribute type="Type">Value</attribute>
```

is written to the output file.

Type is made up of the table name, the data field name, and an optional text field name. **Value** is the value extracted from the corresponding data field. All values are written in text form.

Optionally, instead of the direct data field value, the value of the referenced table (**textref** XML element) can be extracted. The optional **langfieldname** specification extracts the language-specific text of the data field.

Supplementary information is extracted from the data tables using primary key relations.

XML element	XML attribute	Description
fieldtoread	name	Name of the table column containing the data field to be extracted
textref	tablename	Table name of the referenced data table
	reffieldname	Name of foreign key
	textfieldname	Name of the table column containing the data field to be extracted
	langfieldname (optional)	Name of the table column containing the language-specific data field to be extracted
fkpart	readfrom (optional)	Direction of the substring forming the foreign key Valid values: **left**, **right** Default value: **left**

XML element	XML attribute	Description
	startposition	Position from which the substring is formed
	length (optional)	Length of substring Default value: Start or end of the string (**readfrom** XML attribute)

Before data is actually extracted, a check is run as to whether the configured tables and data fields exist and whether the specified system user has appropriate access authorization.

Example

The extracted field values are written to the output file as attributes of a system event.

Unlocalized table fields:

```
...
  <attribute type="VBAP-WERKS">SB</attribute>
...
```

Localized table fields:

```
...
  <attribute type="VBAP-WERKS-NAME">Saarbrücken</attribute>
...
```

5.3.6.1 Handling of NULL values

When handling null values during extraction, it is important to distinguish whether the NULL value or an empty string is extracted. Not every database system used supports differentiation between NULL and empty strings:

- The Oracle database system does not differentiate between no value (NULL) and an empty string. In both cases, no system attribute is created when corresponding database fields are extracted.
- The IMB DB2 and SQL Server database systems do differentiate between NULL and an empty string. If a database field returns NULL as a value during extraction, no system event attribute is created. By contrast, if the system extracts a database field containing an empty string the corresponding system event attribute is created in the XML log file without value:

```
...
  <attribute type="AT_EXAMPLE"></attribute>
...
```

If you want to use default values such as **Not specified** instead of extracted null values after import into the PPM system, you can define default values (**defaultvalue** XML element) in the calculation of the corresponding attribute (see **PPM Customizing** manual).

5.3.6.2 Extract multi-valued fields

Appropriate configuration of the relevant **table** element in the table configuration enables all different values of a field to be extracted (**fieldtoread**) to be written to a system event. A separate class must be used to do this.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jdbc_tableconfiguration SYSTEM
                      'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="MULTIPLE_VALUES">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.jdbc2ppm.
                      ZTableMultipleValues_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
      <fieldtoread name="WMENG"/>
      <fieldtoread name="EDATU"/>
    </table>
    <table name="VBAK" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.jdbc2ppm.
                      ZTableMultipleValues_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBEP"
                              fkfieldname="VBELE"/>
      <fieldtoread name="VBTYP"/>
    </table>
  </configuration>
</jdbc_tableconfiguration>
```

In the example configuration, using the **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableMultipleValues_jdbc2ppm** class for all data tables (**table** elements) specifies that all fields to be extracted (**fieldtoread** elements) are extracted with multiple values if multiple values exist for a field in the source system database. In the XML output file, the extracted field with multiple values appears as an attribute with the same name in a system event:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004969</attribute>
    <attribute type="VBEP-EDATU">19970103</attribute>
    <attribute type="VBEP-EDATU">19970107</attribute>
    <attribute type="VBEP-WMENG">138.000</attribute>
    <attribute type="VBEP-WMENG">317.000</attribute>
    <attribute type="VBEP-WMENG">496.000</attribute>
  </event>
</eventlist>
```

5.3.7 Table access configuration

XML element	XML attribute	Description
xmlextractor_table-configuration		List of R/3 table configurations
configuration	name	Unique identifier of the table configuration
	printname (optional)	Switch. **yes** means that the configuration name is output as a prefix to the field names. Default value: **no**
	classtouse (optional)	Name of the Java class that is used for editing the configuration Default value: Standard implementation
globaltable (optional)	name	Identifier of the table from which global system attributes are extracted
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute

XML element	XML attribute	Description
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
docspec	-	Grouping of the specifications of all document header tables and the document table
docreftable	name	Identifier of a document header table
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
doctable	name	Identifier of a document table. Each line extracted generates a system event in the XML output file.
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation
table (optional)	name	Identifier of the data table from which supplementary information is extracted

XML element	XML attribute	Description
	tablename (optional)	Name of the table in the source system Default value: Specified in the **name** XML attribute
	classtouse (optional)	Name of the Java class that is used for extracting data from the table Default value: Standard implementation

Each SAP table may only be referenced once in the configuration. In order to be able to extract a table multiple times, reference the same table (**tablename** XML attribute) using different identifiers.

Example

You want to extract the **VBAP-MATNR_1** and **VBAP-MATNR_2** fields from the **PRODH** table. The file extract below illustrates the configuration:

```
<table name="PROD_HIER_1" tablename="PRODH" >
  <pkfield name="HNUM" fktablename="VBAP"
                        fkfieldname="MATNR_1"/>
  <fieldtoread name="HVAL"/>
</table>

<table name="PROD_HIER_2" tablename="PRODH" >
  <pkfield name="HNUM" fktablename="VBAP"
                        fkfieldname="MATNR_2"/>
  <fieldtoread name="HVAL"/>
</table>
```

5.3.8 Example: XML configuration and output file

This chapter illustrates the table configuration of an SQL database system by means of a practical example and shows a possible XML output file.

5.3.8.1 Table configuration

The file below shows the **SD_C** configuration for an SQL database system:

```
...
<configuration name="SD_C">
  <docspec>
```

```
  <docreftable name="VBAK">
    <booleancondition>
      <condition fieldname="ERDATCHAR#-#ERZETCHAR"
             logicaloperator="char_creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="ERDAT#-#ERZET"
             logicaloperator="date_creationtimestamp">
        <value>DATE</value>
        <value>TIME</value>
      </condition>
      <condition fieldname="VBTYP" logicaloperator="eq">
        <value>C</value>
      </condition>
    </booleancondition>
    <pkfield name="VBELN" />
  </docreftable>
  <doctable name="Orders" tablename="VBAP">
    <pkfield name="VBELN" fktablename="VBAK"
                     fkfieldname="VBELN"/>
    <pkfield name="POSNR" />
    <fieldtoread name="VBELN"/>
    <fieldtoread name="POSNR"/>
    <fieldtoread name="ERDAT"/>
    <fieldtoread name="ERZET"/>
    <fieldtoread name="MATNR">
      <textref tablename="MAKT" reffieldname="MATNR"
          textfieldname="MAKTX" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="KONDM">
      <textref tablename="T178T" reffieldname="KONDM"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="SPART">
      <textref tablename="TSPAT" reffieldname="SPART"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="WERKS">
      <textref tablename="T001W" reffieldname="WERKS"
                           textfieldname="NAME1"/>
    </fieldtoread>
    <fieldtoread name="CHARG"/>
    <fieldtoread name="PSTYV"/>
    <fieldtoread name="ERNAM"/>
    <fieldtoread name="NETWR"/>
  </doctable>
</docspec>
<table name="VBAK">
  <pkfield name="VBELN" fktablename="Orders"
                              fkfieldname="VBELN"/>
  <fieldtoread name="VTWEG">
    <textref tablename="TVTWT" reffieldname="VTWEG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKORG">
```

```
      <textref tablename="TVKOT" reffieldname="VKORG"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VDATU"/>
    <fieldtoread name="VKBUR">
      <textref tablename="TVKBT" reffieldname="VKBUR"
            textfieldname="BEZEI" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VKGRP">
      <textref tablename="TVGRT" reffieldname="VKGRP"
            textfieldname="BEZEI" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="VBTYP"/>
    <fieldtoread name="AUART"/>
  </table>
  <table name="MARA">
    <pkfield name="MATNR" fktablename="Orders"
                                fkfieldname="MATNR"/>
    <fieldtoread name="MATNR"/>
    <fieldtoread name="MTART">
      <textref tablename="T134T" reffieldname="MTART"
            textfieldname="MTBEZ" langfieldname="SPRAS"/>
    </fieldtoread>
  </table>
</configuration>
```

FOREIGN KEY TABLE

```
...
<docreftable name="VBAK">
  <booleancondition>
    <condition fieldname="ERDATCHAR#-#ERZETCHAR"
              logicaloperator="char_creationtimestamp">
      <value>yyyyMMdd</value>
      <value>HHmmss</value>
    </condition>
    <condition fieldname="ERDAT#-#ERZET"
              logicaloperator="date_creationtimestamp">
      <value>DATE</value>
      <value>TIME</value>
    </condition>
    <condition fieldname="ERDAT_DATE"
                          logicaloperator="date_geq">
      <value>05.12.2001</value>
    </condition>
    <condition fieldname="ERZET_TIME"
                           logicaloperator="time_gt">
      <value>22:27:13</value>
    </condition>
    <condition fieldname="VBTYP"
                               logicaloperator="eq">
      <value>C</value>
    </condition>
  </booleancondition>
  <pkfield name="VBELN" />
</docreftable>
```

...

The identifier and name of the foreign key table from the SQL database system is **VBAK**. All order documents of the **VBTYP=C** type are extracted. The **value** elements of the **char_creationtimestamp** condition operator specify the format of the time stamp to be generated from the corresponding database fields. The **value** elements of the **date_creationtimestamp** condition operator specify the data types of the fields extracted. The values in the **ERDAT** table column are of the **DATE** type and those in the **ERZET** column are of the **TIME** type.

Only those rows in the table in which the values of the **ERDAT_DATE** and **ERZET_TIME** attributes are greater than/equal to or greater than the values specified with **value** (**05.12.2001** or **22:27:13**).

The name of the primary key table column is **VBELN**. For each different **VBELN** field value, data records are extracted from the linked system event table for which **VBELN** has the same value as in the foreign key table.

SYSTEM EVENT TABLE

```
...
<doctable name="Orders" tablename="VBAP">
  <pkfield name="VBELN" fktablename="VBAK"
                              fkfieldname="VBELN"/>
  <pkfield name="POSNR" />
  <fieldtoread name="VBELN"/>
  <fieldtoread name="POSNR"/>
  <fieldtoread name="ERDAT"/>
  <fieldtoread name="ERZET"/>
  <fieldtoread name="MATNR">
    <textref tablename="MAKT" reffieldname="MATNR"
           textfieldname="MAKTX" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="KONDM">
    <textref tablename="T178T" reffieldname="KONDM"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="SPART">
    <textref tablename="TSPAT" reffieldname="SPART"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="WERKS">
    <textref tablename="T001W"
            reffieldname="WERKS" textfieldname="NAME1"/>
  </fieldtoread>
  <fieldtoread name="CHARG"/>
  <fieldtoread name="PSTYV"/>
  <fieldtoread name="ERNAM"/>
  <fieldtoread name="NETWR"/>
</doctable>
...
```

The **VBAP** system event table is assigned the **Orders** identifier. The foreign key relation specified in the line

<pkfield name="VBELN" fktablename="VBAK"
fkfieldname="VBELN"/>

links the table to the **VBAK** foreign key table. The primary key consists of the **VBELN** and **POSNR** columns.

For each table row extracted, a system event (**event** XML element) is created in the XML output file. For each **fieldtoread** XML element, a line of the form

```
<attribute type="...">...</attribute>
```

is written to the output file. For some **fieldtoread** elements, the value extracted from the referenced table (**textref** XML element) is written in addition to the extracted data field value. The optional **langfieldname** extracts the language-specific text of the data field.

The complete source system attribute type is made up of the identifier of the system event table (**doctable name**), the field name (**fieldtoread name**) and the name of the referenced text field (**textref ... textfieldname**). The source system attribute type for the first extracted **fieldtoread** element with a referenced table looks like this:

<attribute type="Orders-MATNR-MAKTX">...</attribute>

DATA TABLE

```
...
<table name="VBAK">
  <pkfield name="VBELN" fktablename="Orders"
                              fkfieldname="VBELN"/>
  <fieldtoread name="VTWEG">
    <textref tablename="TVTWT" reffieldname="VTWEG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKORG">
    <textref tablename="TVKOT" reffieldname="VKORG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VDATU"/>
  <fieldtoread name="VKBUR">
    <textref tablename="TVKBT" reffieldname="VKBUR"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKGRP">
    <textref tablename="TVGRT" reffieldname="VKGRP"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
```

```
  <fieldtoread name="VBTYP"/>
  <fieldtoread name="AUART"/>
</table>
<table name="MARA">
  <pkfield name="MATNR" fktablename="Orders"
                              fkfieldname="MATNR"/>
  <fieldtoread name="MATNR"/>
  <fieldtoread name="MTART">
    <textref tablename="T134T" reffieldname="MTART"
           textfieldname="MTBEZ" langfieldname="SPRAS"/>
  </fieldtoread>
</table>
...
```

The foreign key relations to the **VBELN** and **MATNR** primary key fields in the **Orders** table are used to extract supplementary information from the **VBAK** and **MARA** data tables.

5.3.8.2 XML output file (PPM system event format)

The file extract below shows a range of system events from the XML output file generated using the configuration file specified in the chapter on **Table configuration** (page 102):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="Orders-CHARG"/>
    <attribute type="Orders-ERDAT">20000214</attribute>
    <attribute type="Orders-ERNAM">HDM</attribute>
    <attribute type="Orders-ERZET">143616</attribute>
    <attribute type="Orders-KONDM"/>
    <attribute type="Orders-MATNR">P-100</attribute>
    <attribute type="Orders-MATNR-MAKTX">
            Thread rod M'8 DIN '8895' without tolerance
    </attribute>
    <attribute type="Orders-NETWR">28.70</attribute>
    <attribute type="Orders-POSNR">000010</attribute>
    <attribute type="Orders-PSTYV">TAN</attribute>
    <attribute type="Orders-SPART">01</attribute>
    <attribute type="Orders-SPART-VTEXT">Product category 01
    </attribute>
    <attribute type="Orders-VBELN">0000000001</attribute>
    <attribute type="Orders-WERKS">1000</attribute>
    <attribute type="Orders-WERKS-NAME1">
Plant 1000 (Hamburg)
    </attribute>
    <attribute type="MARA-MATNR">P-100</attribute>
    <attribute type="MARA-MTART">HAWA</attribute>
    <attribute type="MARA-MTART-MTBEZ">
                  Trading goods
    </attribute>
    <attribute type="VBAK-AUART">TA</attribute>
    <attribute type="VBAK-VBTYP">C</attribute>
```

```
    <attribute type="VBAK-VDATU">20000214</attribute>
    <attribute type="VBAK-VKBUR"/>
    <attribute type="VBAK-VKGRP"/>
    <attribute type="VBAK-VKORG">1000</attribute>
    <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
    </attribute>
    <attribute type="VBAK-VTWEG">10</attribute>
    <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
    </attribute>
  </event>
  <event>
    <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
    </attribute>
    <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
    </attribute>
    <attribute type="Orders-SPART">01</attribute>
    <attribute type="Orders-PSTYV">TAD</attribute>
    <attribute type="Orders-MATNR">SERVICE</attribute>
    <attribute type="Orders-ERDAT">20000214</attribute>
    <attribute type="VBAK-VKBUR"/>
    <attribute type="VBAK-VTWEG">10</attribute>
    <attribute type="Orders-SPART-VTEXT">Product category 01
    </attribute>
    <attribute type="Orders-ERZET">143616</attribute>
    <attribute type="VBAK-VKORG">1000</attribute>
    <attribute type="Orders-POSNR">000020</attribute>
    <attribute type="MARA-MTART-MTBEZ">Service
    </attribute>
    <attribute type="Orders-WERKS-NAME1">
Plant 1000 (Hamburg)
    </attribute>
    <attribute type="Orders-CHARG"/>
    <attribute type="Orders-WERKS">1000</attribute>
    <attribute type="VBAK-VDATU">20000214</attribute>
    <attribute type="Orders-MATNR-MAKTX">
                      Repair
    </attribute>
    <attribute type="Orders-VBELN">0000000001</attribute>
    <attribute type="VBAK-AUART">TA</attribute>
    <attribute type="VBAK-VKGRP"/>
    <attribute type="Orders-NETWR">179.00</attribute>
    <attribute type="MARA-MTART">DIEN</attribute>
    <attribute type="VBAK-VBTYP">C</attribute>
    <attribute type="Orders-ERNAM">HDM</attribute>
    <attribute type="MARA-MATNR">SERVICE</attribute>
    <attribute type="Orders-KONDM"/>
  </event>
  <event>
    <attribute type="VBAK-VKGRP-BEZEI">GR. F2 Mr. Mayer
    </attribute>
    <attribute type="VBAK-VKORG-VTEXT">Germany Frankfurt
    </attribute>
    <attribute type="VBAK-VTWEG-VTEXT">Consumer sales
    </attribute>
    <attribute type="Orders-SPART">01</attribute>
    <attribute type="Orders-PSTYV">TAN</attribute>
```

```
    <attribute type="Orders-MATNR">P-100</attribute>
    <attribute type="Orders-ERDAT">20000214</attribute>
    <attribute type="VBAK-VKBUR">1000</attribute>
    <attribute type="VBAK-VTWEG">10</attribute>
    <attribute type="Orders-SPART-VTEXT">Product category 01
    </attribute>
    <attribute type="Orders-ERZET">131257</attribute>
    <attribute type="VBAK-VKBUR-BEZEI">Frankfurt office
    </attribute>
    <attribute type="VBAK-VKORG">1000</attribute>
    <attribute type="Orders-POSNR">000010</attribute>
    <attribute type="MARA-MTART-MTBEZ">
                  Trading goods
    </attribute>
    <attribute type="Orders-WERKS-NAME1">
Plant 1000 (Hamburg)
    </attribute>
    <attribute type="Orders-CHARG"/>
    <attribute type="Orders-WERKS">1000</attribute>
    <attribute type="VBAK-VDATU">20000214</attribute>
    <attribute type="Orders-MATNR-MAKTX">
          Thread rod M'8 DIN '8895' without tolerance
    </attribute>
    <attribute type="Orders-VBELN">0000000002
     </attribute>
    <attribute type="VBAK-AUART">TA</attribute>
    <attribute type="VBAK-VKGRP">101</attribute>
    <attribute type="Orders-NETWR">28.70</attribute>
    <attribute type="MARA-MTART">HAWA</attribute>
    <attribute type="VBAK-VBTYP">C</attribute>
    <attribute type="Orders-ERNAM">HDM</attribute>
    <attribute type="MARA-MATNR">P-100</attribute>
    <attribute type="Orders-KONDM"/>
  </event>
  ...
</eventlist>
```

5.3.9 Block extraction

You can influence the relationship between the execution and memory efficiency of the extraction operation.

The value of the **-cpd** command line argument determines the number of system events that are simultaneously held in the system memory. The maximum possible number depends on the size of the available system memory.

The actual system memory used depends on the following factors:

- Total number of system events to be extracted in the defined analysis period
- Number of system events held in parallel in the system (concurrently processed documents)

- Number of data fields to be extracted (**fieldtoread** and **reftext** XML elements).
- Memory requirements of extracted data fields

5.3.10 Extract the first or last line in a sorting

The Java class to be used **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithInteger_jdbc2ppm** sorts records in a data table by means of the specified sorting criterion, and either writes the first or last record to the system events to be generated. The class must be specified in the **classtouse** attribute of the **table** XML element.

Example

With this class, you can use an integer criterion (amount of items) to sort data records that are referenced via foreign key relations in the **ORDER ITEM** table. The sort criterion is the **AMOUNT** field. From the data sorted, the data record with the largest sorting criterion value (for **ITEM AMOUNT_MAX**) and the data record with the smallest sorting criterion value (for **ITEM AMOUNT_MIN**) is written to the generated system events, each with the corresponding item number. The following file extract illustrates the extraction configuration under the conditions described above:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jdbc_tableconfiguration SYSTEM ↵
                  'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="SortWithInteger">
    <docspec>
      <doctable name="ORDER HEADER">
        <pkfield name="NUMBER"/>
      </doctable>
    </docspec>
    <table name="ITEM AMOUNT_MAX" ↵
          tablename="ORDER ITEM" ↵
          classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                    extractor.jdbc2ppm. ↵
                    ZSortWithInteger_jdbc2ppm">
      <parameter name="SORTCRITERION">
        <value>AMOUNT</value>
      </parameter>
      <parameter name="USE">
        <value>MAX</value>
      </parameter>
      <pkfield name="NUMBER" fktablename="ORDER HEADER" ↵
                                fkfieldname="NUMBER"/>
      <fieldtoread name="ITEM"/>
      <fieldtoread name="AMOUNT"/>
    </table>
    <table name="ITEM AMOUNT_MIN" ↵
          tablename="ORDER ITEM" ↵
          classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                    extractor.jdbc2ppm. ↵
```

```
                    ZSortWithInteger_jdbc2ppm">
      <parameter name="SORTCRITERION">
        <value>AMOUNT</value>
      </parameter>
      <parameter name="USE">
        <value>MIN</value>
      </parameter>
      <pkfield name="NUMBER" fktablename="ORDER HEADER" ↵
                                fkfieldname="NUMBER"/>
      <fieldtoread name="ITEM"/>
      <fieldtoread name="AMOUNT"/>
    </table>
  </configuration>
</jdbc_tableconfiguration>
```

A system event extracted with this configuration could look like this:

```
<event>
  <attribute type="ORDER HEADER NUMBER">
    4711
  </attribute>
  <attribute type="ITEM AMOUNT_MAX-ITEM">
    20
  </attribute>
  <attribute type="ITEM AMOUNT_MAX-AMOUNT">
    100000
  </attribute>
  <attribute type="ITEM AMOUNT_MIN-ITEM">
    50
  </attribute>
  <attribute type="ITEM AMOUNT_MIN-AMOUNT">
    470
  </attribute>
</event>
```

The sorting criterion (in this case **AMOUNT**) is automatically included in the system event specification.

The template for creating the shown configuration looks like this:

```
<table name="..." tablename="..." ↵
      classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                extractor.jdbc2ppm. ↵
                ZSortWithInteger_jdbc2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ...
  See chapter
  JDBC table configuration (page 77)
  ...
</table>
```

The table below reiterates the most important configuration entries for the above **table** definition:

XML element/attribute	Value: Description
name	The specified name is added to the extracted source system attributes as a prefix
tablename (optional)	Table from which the information is to be extracted. Default value: Value from **name** XML attribute
classtouse	**com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithInteger_jdbc2ppm**: Java class to be used
parameter name	
SORTCRITERION	First parameter to be specified. A single **value** XML element must be specified (field name of sorting criterion). The specified field may only contain integer values.
USE	Second parameter to be specified. A single **value** XML element must be specified. Valid values: **MIN** (the data record with the lowest integer value for the sorting criterion is selected) **MAX** (the data record with the highest integer value for the sorting criterion is selected) Either the first or last data record from the list generated based on the specified sorting criterion is selected to be extracted.

You should use this class for extracting the first or last line of a sorting sparingly, as the sorting and selection operations may result in a loss of performance and memory.

5.3.11 Extract the first or last row using a time stamp

You can use the Java class **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithTimestamp_jdbc**

2ppm to extract data by time stamp (for example, Oracle data type **TIMESTAMP**) in a sorted manner. The first or last record extracted is written to an event. The class is specified in the XML element **table** of the XML attribute **classtouse**. The two required parameters are specified in the corresponding XML elements:

- Parameter **name="SORTCRITERION"** with an XML element **value** whose value determines the field name of the table to be extracted.
- Parameter **name="USE"** with an XML element **value** that must have either the value **MAX** (record with the latest value is transferred) or **MIN** (record with the earliest record is transferred).

The following example shows an extract from such a table configuration.

```
<table ...
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortW
ithTimestamp_jdbc2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ... conditions ...
  ... pkfields ...
  ... fieldstoread ...
</table>
```

The specified sort criterion is automatically added to the event output.

Example

Fields of the last process step status on each process step are to be extracted from the table **DBO.WMPROCESSSTEP** (status history of process steps). Depending on the value of the field **AUDITTIMESTAMP** (Oracle data type TIMESTAMP), the latest entry (**MAX**) is to be transferred.

```
<configuration name="AUFK_JCDS_AUFK">
...
  <table name="WMPROCESSSTEP_END" tablename="DBO.WMPROCESSSTEP"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortW
ithTimestamp_jdbc2ppm">
    <parameter name="USE">
      <value>MAX</value>
    </parameter>
    <parameter name="SORTCRITERION">
      <value>AUDITTIMESTAMP</value>
    </parameter>
    <pkfield name="INSTANCEID" fktablename="WMPROCESSSTEP"
fkfieldname="INSTANCEID" />
    <pkfield name="INSTANCEITERATION" fktablename="WMPROCESSSTEP"
fkfieldname="INSTANCEITERATION" />
    <pkfield name="STEPID" fktablename="WMPROCESSSTEP" fkfieldname="STEPID"
/>
    <pkfield name="STEPITERATION" fktablename="WMPROCESSSTEP"
fkfieldname="STEPITERATION" />
    <fieldtoread name="AUDITTIMESTAMP" />
```

```
    <fieldtoread name="INSERTTIMESTAMP" />
    <fieldtoread name="STATUS" />
 </table>
...
</configuration>
```

An event extracted with this configuration could look like this:

```
<event>
...
  <attribute type="WMPROCESSSTEP_END-AUDITTIMESTAMP">24.08.2010
09:51:13.477</attribute>
  <attribute type="WMPROCESSSTEP_END-INSERTTIMESTAMP">24.08.2010
09:51:13.550</attribute>
  <attribute type="WMPROCESSSTEP_END-STATUS">2</attribute>
 ...
</event>
```

EXTRACT DATA INCL. MILLISECONDS OR NANOSECONDS

By default, time stamps are extracted and sorted to the second. If several records with identical time stamp value exist one of them is randomly selected and used for value determination.

If time stamps are saved to the millisecond or to the nanosecond in the database, they can be extracted and sorted to the millisecond respectively nanosecond. To implement this, you can specify the value **MILLISECOND** or **NANOSECOND** in the **precisionoftime** attribute of the XML element **databasesettings** of the JDBC configuration file. The default value is **SECOND**.

With the value **SECOND**, all time stamps or times are extracted in the format **MM/dd/yyyy HH:mm:ss** or **HH:mm:ss** and written to the event file. The data is sorted to the second according to the time stamp.

With the value **MILLISECOND**, time stamps or times are extracted in the format **MM/dd/yyyy HH:mm:ss.SSS** or **HH:mm:ss.SSS** and written to the event file. The data is sorted to the millisecond according to the time stamp.

With the value **NANOSECOND**, time stamps or times are extracted in the format **MM/dd/yyyy HH:mm:ss.nnnnnnnnn** or **HH:mm:ss.SSS** and written to the event file. The data is sorted to the nanosecond according to the time stamp.

The default value is **SECOND**, that is, times are extracted to the second after upgrading an existing client configuration.

The following example shows an extract from the XML JDBC configuration file:

```
<jdbcconf>
  <databasesettings name="jdbc_oracle" dbtype="ORACLE"
precisionoftime="MILLISECOND">
    <driverclass>oracle.jdbc.driver.OracleDriver</driverclass>
    <maxconditionlength>2000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databaseconnection name="jdbc_oracle10_connection">
    <dbsettings>jdbc_oracle</dbsettings>
```

```
    <url>...</url>
    <user>...</user>
...
  </databaseconnection>
</jdbcconf>
```

5.3.12 Multiplication of system events in tables

When extracting from a JDBC database you have the option of creating additional system events based on a system event table.

At the source system level, a 1:n relationship may exist that cannot be resolved when determining the system events. Use the class **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableMultiplyEvents_jdbc2ppm** to create multiple system events from a system event and thus resolve the 1:n relationship.

Example

```
<configuration name="MultiplyEvents">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
         <value>0000006662</value>
         <value>0000006741</value>
       </condition>

       <pkfield name="VBELN" />
      </doctable>
    </docspec>

    <table name="VBAP"
    classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZT
    ableMultiplyEvents_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

    <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

This is the associated system event output file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1460 P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-06</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">MAG PA/DX 175</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-14</attribute>
  <attribute type="VBAP-POSNR">000030</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Jotachi SN4500</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-18</attribute>
  <attribute type="VBAP-POSNR">000040</attribute>
</event>
</eventlist>
```

The following example (excluding the class described) explains the procedure during extraction:

```
<configuration name="MultiplyEvents_Doctable_Only">
   <docspec>
     <doctable name="VBAK" tablename="VBAK">
      <condition fieldname="VBELN" logicaloperator="in">
       <value>0000006662</value>
       <value>0000006741</value>
      </condition>

      <pkfield name="VBELN" />
     </doctable>
   </docspec>

   <table name="VBAP" >
     <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
     <fieldtoread name="POSNR"/>
     <fieldtoread name="MATNR"/>
   </table>
```

```
    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

      <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Since multiple entries in the **VBAP** table are assigned to an entry in the **VBAK** table only one random row is extracted from the **VBAP** table.

This is the associated system event output file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
</eventlist>
```

5.3.13 Extract fields of individual tables for Data analytics

To enable easy data extraction for Data analytics, you can extract the entire contents of a source system table into a file in event format, which can then be imported into a Data analytics analysis realm.

You can also restrict the data to be extractes (page 60) by defining conditions for data extraction.

The table to be extracted is not configured using the table configuration but in the data source file itself. For this, the file **datasource.dtd** contains the following entries:

XML element/attribute	Description
analysistype	XML attribute: Must have the value **DATA_ANALYTICS**.
realmtable	Comprehensive XML element for configuring the Data analytics data source.
tablename	XML attribute of the **realmtable** element: Name of the table in the source system

XML element/attribute	Description
sourcetable	Comprehensive XML element for configuring the Data analytics data source table. Must include at least one **sourcefield** element.
tablename	XML attribute of the **sourcetable** element: Name of the table in the source system
sourcefield	Contains the name of the field of the source system table.

The **analysistype** attribute of the XML element **datasource** must have the value **DATA_ANALYTICS** if the **<realmtable>** element specifies an analysis realm table (default value: PROCESS).

The **tablename** attribute of the **<realmtable>** element indicates the table name of the target table in the analysis realm configuration. The table name does not affect extraction itself, but is evaluated only by the PPM import.

Further information on data import for Data analytics is available in the PPM Data Analytics user guide.

The XML element **<realmtable>** contains the optional element **<sourcetable>** that specifies the table to be extracted. The columns to be extracted from this table must be specified in the **<sourcefield>** element. The element **<sourcetable>** is optional. For the JDBC or SAP Extractor, a single source table and at least one source column must be specified. Otherwise, an error message will be output during the parsing of the data source file.

A data source definition can only have a maximum of one table. It is impossible to restrict the number of rows. All rows are extracted, including all rows with identical values at the columns to be extracted. For example, if the columns **First name** and **Last name** are to be extracted and the table contains ten entries **Peter** and **Schmidt**, ten events with identical attribute values will be generated.

The following example explains the configuration:

```
<realmtable tablename="COMPANY_EMPLOYEE">
   <sourcetable tablename="EMPLOYEE">
      <sourcefield>EMPLOYEE_ID</sourcefield>
      <sourcefield>NAME</sourcefield>
   </sourcetable>
   ...
</realmtable>
<dataextraction>
   <outputfilename>..\custom\testclient\data\employee.xml</outputfilenam
   e>
</dataextraction>
...
<systemconfig>..\custom\testclient\SourceSystemConfig.xml</systemconfig>
```

In contrast to the behavior of the conventional JDCB or SAP Extractor, attributes no longer receive a table name as a prefix in the event output file. For example, if the table **EMPLOYEE** was extracted, the extractor usually generated events of the type **<table name>-<column name>**:

```
<event>
   <attribute type="EMPLOYEE-EMPLOYEE_ID">4711</attribute>
   <attribute type="EMPLOYEE-NAME">Schmidt</attribute>
</event>
```

Extracting a table using the **<realmtable>** element, however, creates only events without table names:

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
```

NULL VALUES IN THE DATA ANALYTICS MODE

If the value of a column is **null** at the time an analysis realm table is extracted, this value is not written to the event. If a row **EMPLOYEE_ID = 4712** exists without last name, the extractor creates the following events.

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
<event>
   <attribute type="EMPLOYEE_ID">4712</attribute>
</event>
```

However, if all column values to be extracted are **null**, the event is written with empty attributes:

```
<event>
   <attribute type="EMPLOYEE_ID">4711</attribute>
   <attribute type="NAME">Schmidt</attribute>
</event>
<event>
   <attribute type="EMPLOYEE_ID"></attribute>
   <attribute type="NAME"></attribute>
</event>
```

If multiple such rows exist they will be – in contrast to the common extraction procedure (analysistype=PROCESS) – transferred accordingly so that as many <event> elements exist in the event file as there are rows in the data table of the source system.

5.3.13.1 Restrict data extraction

If you want to extract only a specific part of the table content, you can define data extraction conditions. You can limit the data volume to be extracted using a time stamp or integer value (for example, a sequence).

To do so, specify the required conditions in the data source configuration using the **condition** element and the **dataextractiontype** attribute.

```
<datasource name="VBAP" type="JDBC" analysistype="DATA_ANALYTICS"
dataextractiontype="TIME_BASED">
      <realmtable tablename="VBAP">
            <sourcetable tablename="VBAP2">
                  <condition logicaloperator="char_creationtimestamp"
fieldname="AEDAT">
                        <value>yyyy-MM-dd</value>
                  </condition>
                  <sourcefield>AEDAT</sourcefield>
                  <sourcefield>ERDAT</sourcefield>
                  <sourcefield>ERNAM</sourcefield>
                  <sourcefield>ERZET</sourcefield>
                  <sourcefield>MATKL</sourcefield>
                  <sourcefield>MATNR</sourcefield>
                  <sourcefield>WERKS</sourcefield>
            </sourcetable>
      </realmtable>
...
</datasource>
```

The **dataextractiontype** attribute can have the following values in Data analytics data sources:

- COMPLETE: Extraction is executed with a time criterion from the date **01.01.1990 00:00:00** and with an integer criterion from the value **0**. This is also the default setting if the attribute does not exist.
- TIME_BASED: A data extraction based on a time stamp is to be run.
- VALUE_BASED: A data extraction based on a value is to be run.

There is no check whether the data extraction condition used matches the data extraction type configured (dataextractiontype=). If the configuration is created using CTK, CTK ensures correct values. If the file is configured manually and a condition that does not match is used, an error may occur during data extraction.

5.3.14 Extract tables with key/value columns

You can use a special extraction class to extract tables that contain cells with key/value pair. A new event attribute is created for each extracted key of an event.

EXTRACTION CLASS

com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableKeyValueFieldsToAttributes_jdbc2ppm

The behavior and configuration of this extraction class is identical to the corresponding extraction class of an SAP system, except for the different class name. For details, see the corresponding chapter Extract tables with key/value columns (page 61).

The following configuration refers to the JDBC extraction class mentioned above:

```
<xmlextractor_tableconfiguration>
    <configuration name="workflow">
        <docspec>
            <doctable name="WF_ITEMS" tablename="WF_ITEMS">
                <pkfield name="ID" />
                <pkfield name="TYPE" />
                <pkfield name="CREATOR" />
                <pkfield name="TEXT" />
            </doctable>
        </docspec>
        <table name="WF_ITEM_ATTRIBUTES" tablename="WF_ITEM_ATTRIBUTES" classtouse=
        "com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableKeyValueFieldsToAttributes_jdbc2ppm">
            <parameter name="KEYVALUEFIELDS">
                <value>ATTRIBUTE#-#VALUE</value>
            </parameter>
            <pkfield name="ID" fktablename="WF_ITEMS" fkfieldname="ID" />
            <fieldtoread name="ATTRIBUTE" />
            <fieldtoread name="VALUE" />
        </table>
    </configuration>
</xmlextractor_tableconfiguration>
```

5.4 Command line program

You create the XML output file(s) using the **runjdbc2ppm.bat** command line program.

Calling up the program without parameters or with **-h** or **-?** outputs the online help on the console, describing all available options.

5.4.1 Command line program arguments

5.4.1.1 General arguments

-VERSION

Outputs the version number of the PPM process extractor used. Other arguments specified are ignored.

-INFORMATION YES|NO

Specify whether information is to be output (yes) or not (no) during the import. The default is **yes**.

-WARNING YES|NO

Specify whether warning messages are to be output (yes) or not (no) during the import. The default is **yes**.

-ERROR YES|NO

This is where you specify whether error messages are to be output (yes) or not (no) during the import. The default is **yes**.

-PROTOCOLFILE <FILE NAME>

Specify the log file to which all messages are written during the import. If you specify a file, only critical error messages resulting in program abortion will be output on the screen.

-LANGUAGE <ISO CODE>

Specify the language in which the log information is to be output.

5.4.1.2 Source database specific arguments

-DATASOURCE <FILE NAME> (OPTIONAL)

Specify the data source you want to use for the extraction. The XML file contains specifications of all files to be used for extraction and XML output. Make sure that you use this parameter instead of and not in combination with the **-datasourcelist**, **-systemconfig**, **-tableconfig**, **-calcconfig**, **-outfile**, or **-nozip** parameters.

-DATASOURCELIST <FILE NAME>

Multiple data sources can be extracted simultaneously by means of the **-datasourcelist** argument. This corresponds to multiple consecutive extraction using the **-datasource** argument. Only JDBC data sources are extracted.

See chapter Extract multiple data sources (page 127).

-SYSTEMCONFIG <FILENAME> <CONFIGNAME>

This is where you specify the name of the XML file containing the system configuration. The content of the file is source database-specific and can contain several configurations. The name of the configuration to be used is specified in the second argument. If the XML file contains only one configuration, this is used automatically. In this case you do not need to specify a configuration name.

-TABLECONFIG <FILE NAME> <CONFIG NAME>

This is where you specify the name of the XML file containing the table configuration. The content of the file is source database-specific. The file must contain all table configurations specified in the table configurations. The name of the table configuration is specified in the second argument. If the XML file contains only one configuration, this is used automatically. In this case you do not need to specify a configuration name.

-CALCCONFIG <FILE NAME> (OPTIONAL)

This is where you specify the name of the XML file with which you can change attributes of the XML output file or add attributes including attribute calculations.

-BEGINDATE <DD.MM.YYYY> (OPTIONAL)

This is where you specify the date from which data is to be extracted from the source database. If you do not use this parameter, the value of the **lastreaddate** XML attribute from the specified system configuration or data source is used. The **yyyymmdd** format is supported.

-BEGINTIME <HH:MM:SS> (OPTIONAL)

Specify the time from which data is to be extracted from the source system. If you do not use this parameter and **-begindate**, the value of the **lastreadtime** XML attribute from the specified system configuration or data source is used. If you do not use the parameter but you do use **-begindate**, the default value **000000** is set.

-ENDDATE <DD.MM.YYYY> (OPTIONAL)

This is where you specify the date up to which data is to be extracted from the source database. If you do not use this parameter, the current date is used. The **yyyymmdd** format is supported.

-ENDTIME <HH:MM:SS> (OPTIONAL)

This is where you specify the time up to which data is to be extracted from the source database. If you do not use the parameter but use **-enddate**, **235959** is set as the default value. If you use neither this parameter nor **-enddate**, the current time is used.

If PPM Process Extractor JDBC-2-PPM is started without specifying a period of time, the start time is extracted from the specified configuration file. The current date and time are used as the end time.

-VALUECONSTRAINT <PARAMETER1> ... <PARAMETER4> (OPTIONAL)

Use this parameter to delimit the data volume to be extracted using an integer criterion. The "<", "<=", ">" and ">=" comparison operators are allowed. You can compare the corresponding

field value with one or two (interval) values. If a last extracted value (**lastreadvalue**) has already been saved in the system configuration or data source, you can also use this value for the comparison (see chapter **Continuous automated extraction** (page 125)).

Examples

```
-valueconstraint 25000020 "<="
Only data records are extracted whose integer value for the field specified
in the event specification is smaller than or equal to the value 25000020.

-valueconstraint 15000020 ">" 25000020 "<"
```

Interval: Only data records are extracted whose integer value for the field specified in the event specification is **smaller than** the value **25000020** and **greater than** the value **15000020**.

-SAVE_VALUE_MINIMUM (OPTIONAL)

Use this parameter to specify that with **-valueconstraint**, the smallest value last read is saved as **lastreadvalue** in the configuration file used (system configuration or data source) when extracting with restrictions. Default value: Saving the largest value

-CPD <INT> (OPTIONAL)

This parameter is used to specify the number of system events to be extracted simultaneously (concurrently processed documents). The specified value gradually extracts the primary key fields from the table referenced using the **doctable** XML element in the table configuration, and stores them in the extractor's Java memory. Specifying a higher value results in improved extraction performance and more efficient memory usage.
If you do not specify the parameter, the recommended default value of 20000 is used.

-PING (OPTIONAL)

This parameter is used to test the connection to the specified source database. No data is extracted.

-REMOVEEMPTY (OPTIONAL)

You use this parameter to remove extracted attributes with no values before the attribute transformation.

5.4.1.3 Output file-specific arguments

-OUTFILE <FILE NAME>

You use this parameter to specify the name of the XML output file. The file extension is added automatically. By default, the file is output as a ZIP file.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

You use this parameter to specify the encoding of the XML output file. Default value: UTF-8

-NOZIP (OPTIONAL)

You use this parameter to specify that the output file is to be output as an XML file rather than as a ZIP file.

-PIKIDATAMAPPING <FILE NAME> <PC NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the measure series in which the process instance-independent measure series are to be saved.

-DIMDATAMAPPING <FILE NAME> <DIM NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the dimension for which the extracted values are subsequently to be imported.

-SORTEVENTATTRIBUTES (OPTIONAL)

You use this parameter to sort the source system attributes in the system events in alphanumeric order using the attribute type. For large data volumes, this parameter can have a negative impact on the extraction speed.

5.4.1.4 Continuous automated extraction

You can extract data in full and automatically by eliminating the **-begindate**, **-begintime**, **-enddate**, **-endtime** parameters in the command line or by using **-valueconstraint** with no value for the first comparison operator. If the data range to be extracted is restricted using integer values, either the smallest or largest of the last values extracted is saved or updated in the configuration file used. If there are time restrictions on the data range to be extracted, the time of the last data extraction is saved or updated in the configuration file.
The following conditions must be met:

- A correct configuration for **char_creationtimestamp** or **valueconstraint** or **date_creationtimestamp** for the data to be extracted is specified in the event specification (see chapter on **Condition operators** (page 89)).
- You always use the same command line call for continuous automated extraction, for example:

```
runjdbc2ppm -datasource datasource.xml -valueconstraint ">"
```

- When you use the **-valueconstraint** parameter, make sure that you either save the greatest of the last values extracted for all extraction operations (default), or the smallest of the last values extracted by specifying **-save_value_minimum**.
  The value of **lastreadvalue** in the configuration file is only updated if the greatest of the last values extracted is greater than the value currently saved when running the command line without the **-save_value_minimum** parameter, or if the smallest of the last values extracted is smaller than the value currently saved when using **-save_value_minimum**.

Example 1 (lastreadvalue with default value "0")

```
-valueconstraint ">="
```

Only those data records are extracted whose integer value for restricting the data range to be extracted is **greater than** or **equal to** the value saved as the last value extracted (**lastreadvalue**) in the system configuration or data source. If no value is saved, the default value **0** is used (in the example, all data records with integer criterion values **>=0** would be extracted). The greatest of the last values extracted (**300** in this example) is saved as **lastreadvalue** according to the default setting (see **-save_value_minimum** parameter).

Example 2 (lastreadvalue="40")

```
-valueconstraint ">=" 270 "<="
```

All data records are extracted that have an integer value of **>=40** and **<=270** for restricting the data range to be extracted. After the extraction, the value for **lastreadvalue** is updated, for example, to **270** if this is really the greatest of the last values extracted.

Example 3 (lastreaddate="19971231" lastreadtime="155959" lastreadvalue="270")

By calling up

```
runjdbc2ppm -datasource datasource.xml -valueconstraint ">"
```

again all data records are extracted whose integer value for delimiting the data range to be extracted is greater than **270** and whose time stamp is as old as or more recent than the start time of the data extraction (**31.12.1997 15:59:59**).

The start time of data extraction is entered as **lastreaddate**/**lastreadtime** in the configuration file.

If you eliminate the **-begindate**/**-begintime** parameters, you always extract the data records whose time stamps are more recent (greater) than the ones specified and last extracted using **lastreaddate**/**lastreadtime**. The current date and time are entered as **lastreaddate**/**lastreadtime** if you do not use the **-enddate** and **-endtime** parameters.

5.5 Extract multiple data sources

You can use a data source list to extract multiple data sources simultaneously. The data source list is specified in a separate configuration file.

Each new JDBC data source created in CTK is automatically added to the end of the list of data sources of the current client. During extraction using the command line option **-datasourcelist <file name>** the data of these data sources is extracted consecutively as if data extraction was called multiple times using the option **-datasource <file name>**. The sequence of the data source extraction is specified in the configuration file. Data sources of a type other than JDBC in the list are ignored during extraction.

The following applies to trouble shooting:

- If the extraction is called via a valid data source list that does not contain any matching data sources the extraction ends without outputting an error message.
- If the extraction is called using a data source list containing multiple matching data sources, and if an error occurs during the extraction of one of these data sources resulting in extraction cancelation, the procedure continues with the next data source. This means that cancelation of the extraction of one data source does not lead to cancelation of the entire procedure.
- If an error occurs in at least one data source, which would have led to an exit error status during the single extraction of this data source, the last exit error status will be returned by the overall data source extraction process.

5.6 Extract data of BIT data type

You can extract data of **BIT** data type from SQL server databases. A value extracted from a BIT-data field can be mapped to a value of BOOLEAN data type.

Example of extracted BIT data fields

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <metadata>
    <attr_desc type="DATATYPE_BIT-ID">
      <ppmdatatype>LONG</ppmdatatype>
    </attr_desc>
    <attr_desc type="DATATYPE_BIT-VALUE_BIT_2">
      <ppmdatatype>BOOLEAN</ppmdatatype>
    </attr_desc>
    <attr_desc type="DATATYPE_BIT_2-VALUE_BIT_1">
      <ppmdatatype>BOOLEAN</ppmdatatype>
```

```
    </attr_desc>
    <attr_desc type="DATATYPE_BIT_2-VALUE_BIT_3">
      <ppmdatatype>BOOLEAN</ppmdatatype>
    </attr_desc>
    <attr_desc type="DATATYPE_BIT_2-VALUE_INT">
      <ppmdatatype>LONG</ppmdatatype>
    </attr_desc>
  </metadata>
<event>
  <attribute type="DATATYPE_BIT-ID">1</attribute>
  <attribute type="DATATYPE_BIT-VALUE_BIT_2">false</attribute>
  <attribute type="DATATYPE_BIT_2-VALUE_BIT_1">false</attribute>
  <attribute type="DATATYPE_BIT_2-VALUE_BIT_3">true</attribute>
  <attribute type="DATATYPE_BIT_2-VALUE_INT">1</attribute>
</event>
```

You can use table fields of **BIT** data type also in conditions to restrict data extraction. The value can be **true** (respectively **1**) or **false** (respectively **0**).

Example of an extraction configuration

```
<jdbc_tableconfiguration>
        <configuration name="BitConfig">
                <docspec>
                        <doctable name="DATATYPE_BIT">
                                <condition fieldname="VALUE_BIT_1" logicaloperator="eq">
                                        <value>true</value>
                                </condition>
                                <pkfield name="ID"/>
                        </doctable>
                </docspec>

                <table name="DATATYPE_BIT">
                        <pkfield name="ID" fktablename="DATATYPE_BIT" pkfieldname="ID"/>
                        <fieldtoread name="VALUE_BIT_2"/>
                </table>

                <table name="DATATYPE_BIT_2" tablename="DATATYPE_BIT">
                        <pkfield name="VALUE_BIT_2" fktablename="DATATYPE_BIT"
fkfieldname="VALUE_BIT_2"/>
                        <fieldtoread name="VALUE_BIT_1"/>
                        <fieldtoread name="VALUE_BIT_3"/>
                        <fieldtoread name="VALUE_INT"/>
```

```
        </table>
    </configuration>
</jdbc_tableconfiguration>
```

5.7 Appendix

5.7.1 Supported data types

The supported database systems include numerous data types that are not contained in all database systems or that have other names or different semantics there.

To avoid undesired behavior, the list below indicates data types that have been tested successfully with PPM Process Extractor JDBC-2-PPM.

Database system	Data types	Note
Oracle	CHAR, VARCHAR2, NCHAR, NVARCHAR2, NUMBER, LONG, DATE, TIMESTAMP	The CLOB, NCLOB, BINARY_FLOAT, and BINARY_DOUBLE data types are not supported.
IBM DB2	VARCHAR, CHAR, BIGINT, REAL, DOUBLE, FLOAT, DATE, TIME, TIMESTAMP	The LONG VARCHAR and CLOB data types are not supported.
Microsoft SQL Server	CHAR, VARCHAR, NCHAR, NVARCHAR, INT, TINYINT, SMALLINT, BIGINT, REAL, FLOAT, DECIMAL, DATETIME, SMALLDATETIME, BIT	The LONG VARCHAR and CLOB data types are not supported.

5.7.2 Special characters and capitalization

From version 9.9, PPM Process Extractor JDBC-2-PPM supports the usage of special characters and case-sensitive capitalization in schema, table, and field names in SQL databases.

SQL expressions must be in brackets so that database systems can process them if they contain special characters in schema, table, and column names and case-sensitive spelling, for example like in the following SQL statement for an SQL server database.

```
SELECT [column with space] FROM [USER.1].[table $%&]
```

Depending on the database system, the syntax may differ. For example, under Oracle and DB2, names must be set in apostrophes, and under SQL server, square brackets can be used.

To support special characters and case-sensitive spelling, the system configuration **JdbcConfig.dtd** contains the **non-standard-sql-identifiers (true | false)** attribute. If the **non-standard-sql-identifiers** attribute has the value **true**, the schema, table, and column names will be set in limiting characters suitable for the relevant database type.

```
<!ATTLIST databasesettings
    name ID #REQUIRED
    dbtype (ORACLE | DB2 | SQLSERVER | OTHER) #REQUIRED
    precisionoftime (SECOND|MILLISECOND) "SECOND"
    non-standard-sql-identifiers (true | false) "false"
```

In the database systems, limiting characters must not be part of database identifiers, such as schema, table, and column names.

BACKWARDS COMPATIBILITY

To ensure compatibility with existing configurations, the value **false** is assigned to the attribute **non-standard-sql-identifiers** in CTK when opening and saving or migrating a configuration. An existing value is not automatically overwritten, but can be changed manually in CTK.

When creating a new configuration, CTK sets this value to **true** by default.

5.7.2.1 Special case: Schema and table name

Extraction of tables located in a database schema other than that of the user logged in is possible in the JDBC extractor only via a prefix of the table name. The prefix is separated by a period from the table name in the **tablename** attribute in the various elements of the table configuration (jdbc_tableconfiguration.dtd).

Example

<doctable name="customer" tablename="USER1.CUSTOMER ">

Since the period can also occur as a special character in schema and table names, each period that is part of the schema or table name must be masked with a backslash ****. To recognize backslashes that are part of the name they must be masked by a double occurrence.

Schema and table names can be unambiguously differentiated in the following example configurations.

Schema name	Table name	Syntax for tablename attribute
USER	1.CUSTOMER	USER.1\.CUSTOMER
USER.1.1	ORDER.DE	USER\.1\.1.ORDER\.DE
USER.	ORDER	USER\..ORDER
USER\2	ORDER.DE	USER\\2.ORDER\.DE
USER\.1	ORDER\.44	USER\\\.1.ORDER\\\.44
USER	1.TABLE.1	USER.1.TABLE.1

If more than one period occurs in a schema and table name (see example in the last table row) which is not masked by the backslash, the first period is interpreted as a schema separator and the other periods are seen as parts of the table name. This corresponds to version pre-9.9 behavior and is kept for compatibility reasons. It is still recommended to mask all periods except the schema separator with a prefix backslash.

Masking any other character than period or backslash is not allowed.

When defining foreign keys it is important to observe that the value in the **<fktablename>** XML attribute refers to the **<name>** XML attribute and not to **<tablename>**.

Example

```
<jdbc_tableconfiguration>
 <configuration name="Example">
   <docspec>
    <doctable name="USER.1.CUSTOMER" tablename="USER\.1.CUSTOMER">
     <condition fieldname="SEQUENCE" logicaloperator="in">
      <value>001</value>
      <value>099</value>
     </condition>
     <pkfield name="SEQUENCE"/>
    </doctable>
   </docspec>
   <table name="FK_TABLE">
    <pkfield name="SEQUENCE"
      fktablename="USER.1.CUSTOMER" fkfieldname="SEQUENCE"/>
    <fieldtoread name="FK_COLUMN1"/>
    <fieldtoread name="FK_COLUMN2"/>
   </table>
 </configuration>
</jdbc_tableconfiguration>
```

If the **tablename** is missing, the contents of the **name** attribute is used as a table name to access the database. The two attribute values for the **name** and **fktablename** attributes are interpreted as pure character strings and therefore not evaluated in terms of separators and masking characters.

For example, if the table is specified only with the **name** attribute like this:

<doctable name="USER\.CUSTOMER">.

this means that

<doctable name="USER\.CUSTOMER" tablename="USER\.CUSTOMER">.

The extractor would thus extract the table **USER.CUSTOMER** from the database.

If special characters occur in the table name we recommend to specify the table name always with the **tablename** attribute.

6 PPM Process Extractor CSV-2-PPM

This chapter provides an overview of the architecture, functioning and configuration of PPM Process Extractor CSV-2-PPM.

6.1 CSV

A file in CSV (Comma Separated Values) format consists of the actual data records and an optional header containing the column headings. The column headings in the header and the data fields for the data records are separated by the same separator, normally a semicolon. A data field may only contain the separator if it is used within a masked value. PPM Process Extractor CSV-2-PPM implements the column headings as attribute types, that is, the values of the data records as attribute values.

Example

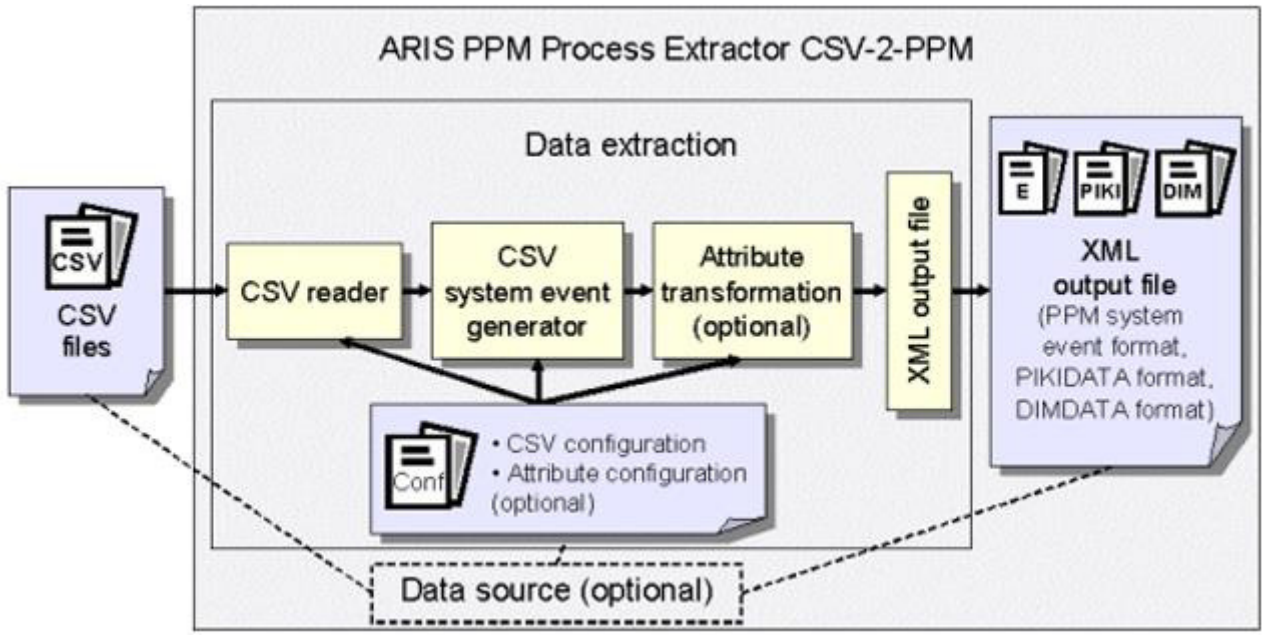
```
ORDER NUMBER,POSITION,RECORDED BY,MATERIAL,QUANTITY
4711,10,"Harry, ""A"" Williams",Mobile 6600,3
4811,23,"Ben, ""B"" Snyder",Mobile 6601,2
4911,15,"George, ""C"" Nyland",Mobile 6602,1
```

The example CSV file contains a header with the column headings for five data columns and three data records.

The comma is used as the separator and double apostrophes are used as the masking character. The masking character is used within a masked data field value by doubling the masking character used. The values in the **RECORDED BY** column may contain the separator, as they are masked.

6.2 Architecture

The figure below illustrates the functionality of PPM Process Extractor CSV-2-PPM:



The CSV files are read using the CSV reader, based on the CSV configuration. The CSV system event generator and, if applicable, the attribute transformation transfer the information extracted into either PPM system event format, PIKIDATA, or DIMDATA format. The XML output writes the data to the corresponding XML output file.

6.3 CSV reader and CSV system event generator

The CSV reader is used to read CSV files. To import CSV files with a header, you must include an appropriate specification in the CSV configuration file, so that the first line of the CSV file is interpreted as a header (see chapter on **CSV configuration** (page 137)).

Capitalization in the column headings and data field values is transferred unchanged from the CSV files.

The following chapters describe the behavior of the CSV reader and the CSV system event generator using simple examples. In the examples, the CSV file is compared to the corresponding XML output file in PPM system event format. A CSV data record and column headings are transformed into a system event in the XML output file in PPM system event format.

6.3.1 Example 1: Complete header

ENTRY IN CSV FILE

```
ORDER NUMBER;POSITION;RECORDED BY;MATERIAL;QUANTITY
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

ENTRY IN XML OUTPUT FILE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
     <event>
     <attribute type="ORDER NUMBER">4711</attribute>
     <attribute type="POSITION">10</attribute>
     <attribute type="RECORDED BY">Harry, "A" Williams</attribute>
     <attribute type="MATERIAL">Mobile 6600</attribute>
     <attribute type="QUANTITY">3</attribute>
     </event>
     ...
</eventlist>
```

If you specify a masking character in the CSV configuration file, all values are unmasked when generating the XML output file.

6.3.2 Example 2: Header with missing column heading

If a column heading is missing in the CSV files, it is automatically added in the XML output file. By default, the column name assigned is made up of the prefix **FIELD_** and the position number of the column. The first column has the position 1.

ENTRY IN CSV FILE

```
ORDER NUMBER;POSITION;;MATERIAL;QUANTITY
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

ENTRY IN XML OUTPUT FILE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="ORDER NUMBER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams
    </attribute>
    <attribute type="MATERIAL">Mobile 6600</attribute>
    <attribute type="QUANTITY">3</attribute>
  </event>
  ...
</eventlist>
```

The column heading **FIELD_3** is automatically generated in place of the missing column heading in the CSV files.

6.3.3 Example 3: Header with several missing column headings

If several column headings are missing in the CSV files, they are automatically added in the XML output file. If the missing column headings are located at the end of the header, the separators do not need to be present between the missing column headings.

ENTRY IN CSV FILE

```
ORDER NUMBER;POSITION;;;
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

or

```
ORDER NUMBER;POSITION
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

ENTRY IN XML OUTPUT FILE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="ORDER NUMBER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams</attribute>
    <attribute type="FIELD_4">Mobile 6600</attribute>
    <attribute type="FIELD_5">3</attribute>
  </event>
  ...
</eventlist>
```

The column headings **FIELD_3**, **FIELD_4** and **FIELD_5** are automatically created in place of the missing column headings in the CSV files. The same entry is generated in the XML output file regardless whether the separator appears between the missing column headings.

6.3.4 Example 4: No header

If the CSV files do not contain a header, a column heading is generated automatically for each data column.

ENTRY IN CSV FILE

```
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

ENTRY IN XML OUTPUT FILE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="FIELD_1">4711</attribute>
```

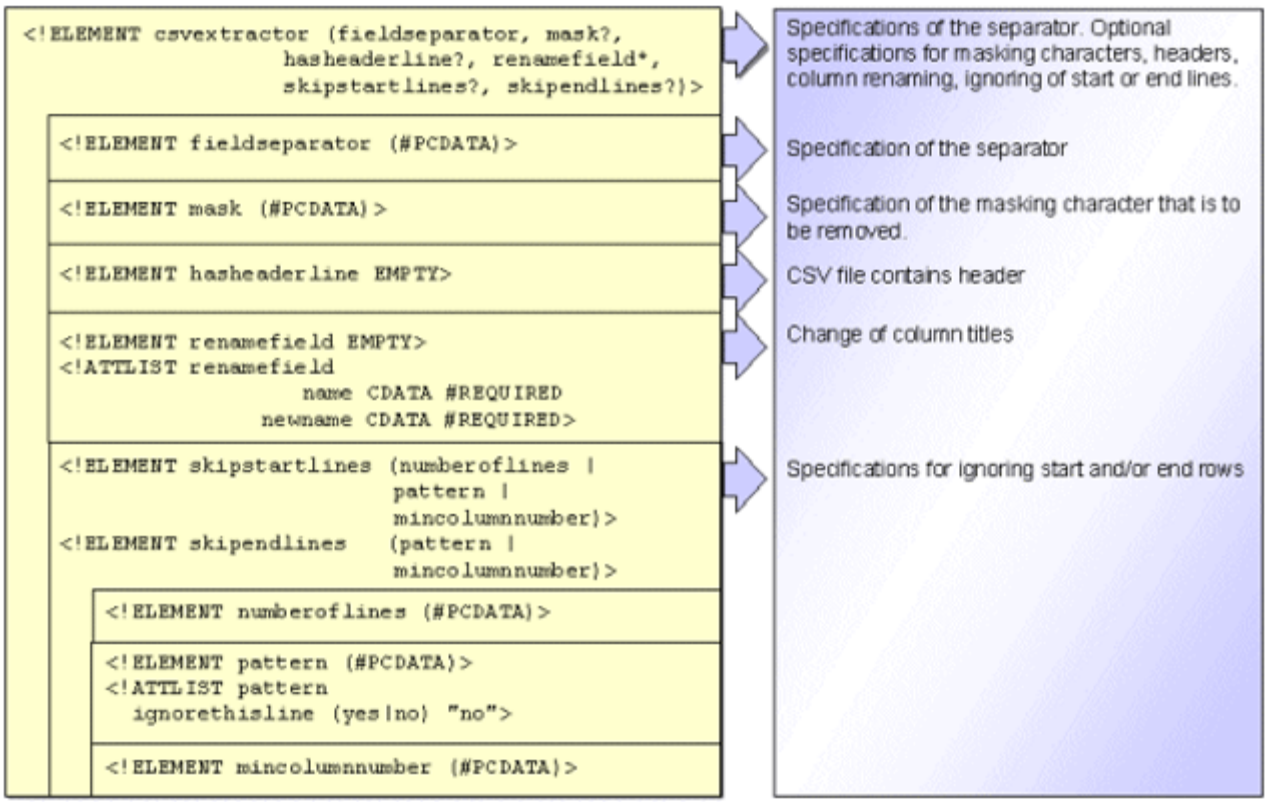
```
    <attribute type="FIELD_2">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams</attribute>
    <attribute type="FIELD_4">Mobile 6600</attribute>
    <attribute type="FIELD_5">3</attribute>
  </event>
  ...
</eventlist>
```

6.4 CSV configuration

The way in which the CSV data is processed when it is imported is specified in an XML configuration file. The name of this file is transferred to the **runcsv2ppm** command line program as an argument.

In the configuration file, you can specify the separator or rename column headings.

The format of the XML file is specified by the following DTD:



XML element or attribute	Description	Example
csvextractor	Configuration of CSV import process	See below
fieldseparator	Separator	,

XML element or attribute	Description	Example
mask	Masking character (values imported are unmasked)	"
hasheaderline	CSV files include header	
renamefield	Rename a data column	See below
name	Name of a data column generated from CSV files	MATERIAL or FIELD_3
newname	New name of data column	Item designation
skipstartlines	Start lines to be ignored	
skipendlines	End lines to be ignored	
numberoflines	Number of start lines to be ignored. Counting begins from the first line.	10
pattern	Character pattern for locating a data line from which lines are to be extracted (**skipstartlines**: by default, the line found is also extracted) or up to which lines are to be extracted (**skipendlines**: line found including all following lines are not extracted The following placeholders are permitted: ***** No or any number of characters **?** Any single character **** Masking character for searching for placeholders or masking characters in the form: **** or ***** or **\?** The first line from the top that matches the specified pattern is selected.	For example, the expression ***10?*269** searches for values such as 555108*269 or 104*269 but not 104\\555269.

XML element or attribute	Description	Example
ignorethisline (only with **pattern** in **skipstartlines**)	Ignore (**yes**) or do not ignore (**no**) the data line containing the desired pattern in the extraction. Default value: **no**	yes
mincolumn number	Specifies a minimum number of columns to define the start and/or end of the data range to be extracted. **skipstartlines**: Lines are extracted from (inclusive) the first line found with at least the specified number of columns. **skipendlines**: Lines are extracted up to (exclusive) the first line found with less than the specified number of columns.	4

EXAMPLE CONFIGURATION (CSVCONFIG.XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE csvextractor SYSTEM 'csvextractor.dtd'>
<csvextractor>
  <fieldseparator>;</fieldseparator>
  <mask>"</mask>
  <hasheaderline/>
  <renamefield name="FIELD_3" newname=
                       "Order item recorded by"/>
  <renamefield name="MATERIAL" newname="Item designation"/>
  <skipstartlines>
    <pattern ignorethisline="yes">###*</pattern>
  </skipstartlines>
  <skipendlines>
    <pattern>5???;*;*;*;*</pattern>
  </skipendlines>
</csvextractor>
```

CSV FILE TO BE IMPORTED (EXAMPLE.CSV)

The file consists of a comment line, a header, and five data records:

```
### Order data 03/25/2006 ###
ORDER NUMBER;POSITION;;MATERIAL;QUANTITY
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
4811;23;"Ben, ""B"" Snyder";Mobile 6601;2
4911;15;"George, ""C"" Nyland";Mobile 6602;1
5011;6;"George, ""C"" Nyland";Mobile 5405;2
5211;23;"George, ""C"" Nyland";Mobile 5410;1
```

OUTPUT FILE IN PPM SYSTEM EVENT FORMAT (EVENT.XML)

The CSV system event generator uses the example CSV configuration to generate the following XML output file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="ORDER NUMBER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="Order item recorded by"
                >Harry, "A" Williams</attribute>
    <attribute type="Item designation">Mobile 6600
    </attribute>
    <attribute type="QUANTITY">3</attribute>
  </event>
  <event>
    <attribute type="ORDER NUMBER">4811</attribute>
    <attribute type="POSITION">23</attribute>
    <attribute type="Order item recorded by"
                  >Ben, "B" Snyder</attribute>
    <attribute type="Item designation">Mobile 6601
    </attribute>
    <attribute type="QUANTITY">2</attribute>
  </event>
  <event>
    <attribute type="ORDER NUMBER">4911</attribute>
    <attribute type="POSITION">15</attribute>
    <attribute type="Order item recorded by"
                >George, "C" Nyland</attribute>
    <attribute type="Item designation">Mobile 6602
    </attribute>
    <attribute type="QUANTITY">1</attribute>
  </event>
</eventlist>
```

In accordance with the specifications in the CSV configuration file, the following command line call generates one system event in the XML output file from each data record in the CSV data:

```
runcsv2ppm -i example.csv -csvconfig csvconfig.xml -outfile event -nozip
```

When importing, for all system events the name **FIELD_3** assigned to the third data column is automatically replaced with **Order item recorded by** and the **MATERIAL** data column is renamed **Item designation**. The masked values of the **Order item recorded by** attribute type are unmasked. The comment line is ignored in the extraction as specified for **skipstartlines**. The last two lines in the data range are ignored as specified for **skipendlines** (order numbers **5011**, **5211**).

6.4.1 CSV configuration extension

You can specify the CSV input file(s) and the encoding of the input file(s) in the CSV source system configuration. The XML element **csvextractor** can contain the following two XML elements:

XML element/attribute	Description
inputfile	Contains the path of the input file(s)
encoding	Contains the encoding of the input file(s)

Example

```
<csvextractor>
    <inputfile>../testclient/data/customers.csv</inputfile>
    <encoding>ISO-8859-15</encoding>
    <fieldseparator>,</fieldseparator>
    <mask>"</mask>
    <hasheaderline/>
</csvextractor>
```

The value for the XML element **inputfile** can be the path to the CSV file. If you specified the path, the command line parameter (page 142) **-i** will be ignored. The same applies to the XML element **encoding** and the command line parameter (page 142) **-encoding**.

If you specified the path to the CSV file (inputfile) in the CSV configuration, the path data in the data source will be ignored.

6.5 Command line program

The XML output file is created with the **runcsv2ppm.bat** command line program.

Calling up the program without parameters or with **-h** or **-?** outputs the online help on the console. This describes all available options.

6.5.1 Command line program arguments

6.5.2 General arguments

-VERSION

Outputs the version number of the PPM process extractor used. Other arguments specified are ignored.

-INFORMATION YES|NO

Specify whether information is to be output (yes) or not (no) during the import. The default is **yes**.

-WARNING YES|NO

Specify whether warning messages are to be output (yes) or not (no) during the import. The default is **yes**.

-ERROR YES|NO

This is where you specify whether error messages are to be output (yes) or not (no) during the import. The default is **yes**.

-PROTOCOLFILE <FILE NAME>

Specify the log file to which all messages are written during the import. If you specify a file, only critical error messages resulting in program abortion will be output on the screen.

-LANGUAGE <ISO CODE>

Specify the language in which the log information is to be output.

6.5.3 CSV-specific arguments

-DATASOURCE <FILE NAME> (OPTIONAL)

Specify the data source you want to use for the extraction. The XML file contains specifications of all files to be used for extraction and XML output. Make sure that you use this parameter instead of and not in combination with the **-datasourcelist**, **-csvconfig**, **-i**, **-calcconfig**, **-outfile**, or **-nozip** parameters.

-DATASOURCELIST <FILE NAME>

Multiple data sources can be extracted simultaneously by means of the **-datasourcelist** argument. This corresponds to multiple consecutive extraction using the **-datasource** argument. Only CSV data sources are extracted.

See chapter Extract multiple data sources (page 144).

-I <CSVFILENAME> [<CSVFILENAME> ...]

This is where you specify the CSV file(s) to be used for generating an XML output file.

If the path to the CSV input file is specified in the CSV configuration file, the command line parameter **-i** will be ignored.

-CSVCONFIG <FILE NAME>

This is where you specify the name of the CSV configuration file containing the information for generation of the XML output file from the CSV files.

-ENCODING "<ENCODINGNAME>" (OPTIONAL)

This is where you specify the encoding of the CSV files. If you do not specify anything, **ISO-8859-15** is used.

If the encoding is specified in the CSV configuration file, the command line parameter **-encoding** will be ignored.

-CALCCONFIG <FILE NAME> (OPTIONAL)

This is where you specify the name of the XML configuration file with which you can change attributes of the XML output file or add attributes including attribute transformations.

-REMOVEEMPTY (OPTIONAL)

You use this parameter to remove extracted attributes with no values before the attribute transformation.

6.5.4 Output file-specific arguments

-OUTFILE <FILE NAME>

You use this parameter to specify the name of the XML output file. The file extension is added automatically. By default, the file is output as a ZIP file.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

You use this parameter to specify the encoding of the XML output file. Default value: UTF-8

-NOZIP (OPTIONAL)

You use this parameter to specify that the output file is to be output as an XML file rather than as a ZIP file.

-PIKIDATAMAPPING <FILE NAME> <PC NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the measure series in which the process instance-independent measure series are to be saved.

-DIMDATAMAPPING <FILE NAME> <DIM NAME> (OPTIONAL)

You use this parameter to specify the name of the XML file containing the mapping in the first argument. In the second argument, you specify the name of the dimension for which the extracted values are subsequently to be imported.

-SORTEVENTATTRIBUTES (OPTIONAL)

You use this parameter to sort the source system attributes in the system events in alphanumeric order using the attribute type. For large data volumes, this parameter can have a negative impact on the extraction speed.

6.6 Extract multiple data sources

You can use a data source list to extract multiple data sources simultaneously. The data source list is specified in a separate configuration file.

Each new CSV data source created in CTK is automatically added to the end of the list of data sources of the current client. During extraction using the command line option **-datasourcelist <file name>** the data of these data sources is extracted consecutively as if data extraction was called multiple times using the option **-datasource <file name>**. The sequence of the data source extraction is specified in the configuration file. Data sources of a type other than CSV in the list are ignored during extraction.

The following applies to trouble shooting:

- If the extraction is called via a valid data source list that does not contain any matching data sources the extraction ends without outputting an error message.
- If the extraction is called using a data source list containing multiple matching data sources, and if an error occurs during the extraction of one of these data sources resulting in extraction cancelation, the procedure continues with the next data source. This means that cancelation of the extraction of one data source does not lead to cancelation of the entire procedure.
- If an error occurs in at least one data source, which would have led to an exit error status during the single extraction of this data source, the last exit error status will be returned by the overall data source extraction process.

7 Functionalities of all extractors

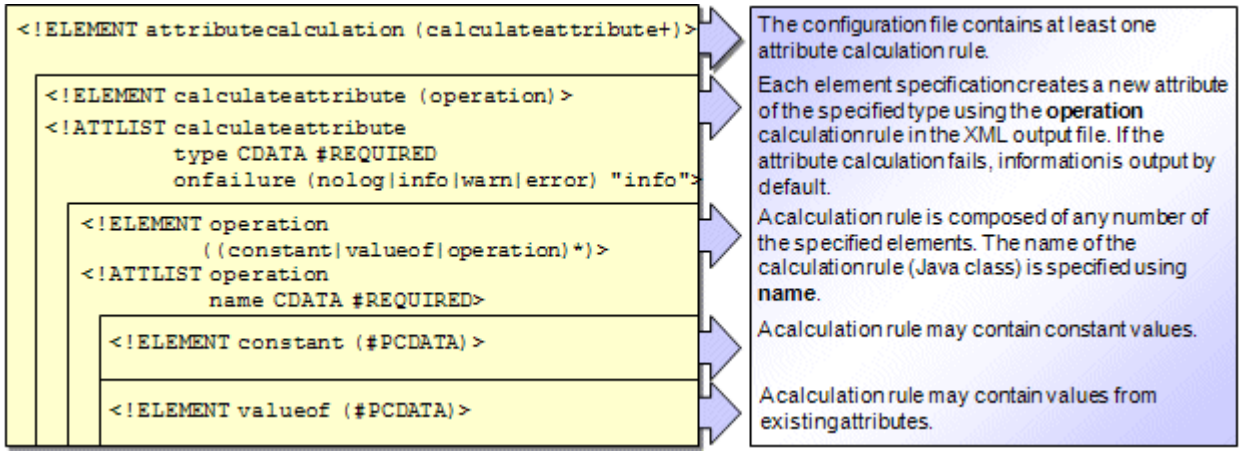
This chapter describes the program features that all of the process extractors have in common.

7.1 Attribute transformation

To change the values of particular attribute types in the output files or to create additional attribute types, you need to specify this in an XML configuration file. The name of this XML configuration file is transferred to the command line program as an argument.

This configuration file contains the settings for processing existing attribute types and for generating and transforming new attribute types when creating the XML output file(s).

The format of the XML file is specified by the following DTD:



XML element or XML attribute	Description
attributecalculation	Configuration of attribute transformation
calculateattribute	Generation of a system event attribute type for successful execution of the transformation rule (**operation**)
type	Name of the system event attribute type generated

XML element or XML attribute	Description
onfailure	Cancelation behavior if transformation fails. Valid values: **nolog** (no log output) **info** (output as information) **warn** (output as warning) **error** (output as error message) Default value: **info**
operation	Transformation rule that is implemented by a Java class. Each operation has a single return value in the form of a string. If the operation fails, nothing is returned.
name	Name of operation
constant	Constant value to be used in the transformation
valueof	Value of an existing or already transformed source system attribute type

The **calculateattribute** elements are processed in the order in which they are specified in the configuration file. Nesting of the **operation** XML elements is processed outwards or from top to bottom.

For reasons of efficiency, during an attribute transformation only those arguments that are essential for the transformation are evaluated. For example, in the **if_then_else** operation, after the truth of the first argument is checked, only the second argument (then) or the third argument (else) is processed. The attribute transformation is canceled with an error message as soon as a required element cannot return a value.

If the **onfailure** XML element in the attribute transformation file has the value **error**, unsuccessful attribute generation results in the corresponding system event being assigned the **EVENT_HAS_ATTRIBUTE_ERRORS=true** attribute in the XML output file. In addition, an error message is generated in the log file.

If the **onfailure** XML attribute has the value **warn**, unsuccessful attribute generation results in the corresponding system event being assigned the **EVENT_HAS_ATTRIBUTE_WARNINGS=true** attribute in the XML output file. In addition, a warning is generated in the log file.

Warning

calculateattribute allows the values of existing attribute types to be overwritten after successful execution of the corresponding transformation rules (**operation**). No warning is output. If necessary, ensure that the values of existing attribute types are not accidentally overwritten by performing a conditional existence check with a suitable transformation rule. You can use the **exists** operation (see chapter **Operations** (page 148)) for this purpose.

If the entire transformation for an attribute is canceled, the existing attribute value remains unchanged. Depending on what you have specified for the **onfailure** option, a corresponding message is output with a reference to the system event generated. For **nolog**, no log is output.

Example

The extraction of source system data from the R/3 system and the subsequent processing by the XML Generator have created the following entry in the **event.xml** XML output file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

Using the following specifications in the **attributetransform.xml** attribute transformation file, you can change the display of the values of the **VBAP-ERNAM** attribute type so that the concatenation of the original value with a constant string will be used as the new value.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE attributecalculation SYSTEM
        "attributetransformation.dtd">
<attributecalculation>
  <calculateattribute type="VBAP-ERNAM">
  <operation name="concat">
  <constant>Walter, "A" </constant>
  <valueof>VBAP-ERNAM</valueof>
  </operation>
  </calculateattribute>
  ...
</attributecalculation>
```

The following command line call creates the desired XML output file from the original XML output file using the attribute transformation file discussed above.

```
runjdbc2ppm -systemconfig SysConfig.xml ides_doc_vdap -tableconfig
TableConf.xml SD_C -begindate 01.01.2005 -enddate 31.01.2005 -calcconfig
attributetransform.xml -outfile event -outfileencoding ISO-8859-1 -nozip
```

The same entry as the one above then looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">Walter, "A" SCHMIDT
    </attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

7.1.1 Operations

The PPM process extractors provide the operations outlined in the following chapters for transformation or modification of attribute types.

For the sake of simplicity, the syntax examples shown for the **operation** XML element exclusively use constants. However, the **valueof** and **operation** elements can also be specified in an operation.

If a value cannot be determined with a transformation rule, the transformation of the attribute type is only canceled if the value not determined is essential for the subsequent transformation. Depending on what is specified for the **onfailure** option, a corresponding message may be output in this case.

The return value of the operations is generally an alphanumeric string.

7.1.1.1 unmask

Description:	Removes the masking of a value
Syntax:	Two parameters: 1. Masking character 2. Value to be unmasked

Example:	`<operation name="unmask">`
	`  <constant>\</constant>`
	
	`<constant>\C:\\ppm3\\ppm320\\build3714\\dtd\`
	`  </constant>`
	`</operation>`

Return value:	**C:\ppm3\ppm320\build3714\dtd** If no value can be determined for one of the parameters, nothing is returned.

The value to be unmasked must begin and end with the specified masking character.

7.1.1.2 trim

Description:	Removes the spaces at the beginning and end of a value
Syntax:	One parameter: Value for which the spaces at the beginning and end are to be removed

Example:	`<operation name="trim">`
	`  <constant> PPM </constant>`
	`</operation>`

Return value:	<...>PPM<...> If no value can be determined for one of the parameters, nothing is returned.
Note:	The number of spaces at the beginning and end of the field value is unimportant.

7.1.1.3 concat

Description:	Concatenation of any number of values
Syntax:	One to n parameters

Example:	`<operation name="concat">`
	`  <constant>Order processing</constant>`
	`  <constant>/</constant>`
	`  <constant>Cash sales</constant>`
	`</operation>`

Return value:	**Order processing/Cash sales** If no value can be determined for one of the parameters, nothing is returned.

7.1.1.4 substring

Description:	Identifies a substring of a value
Syntax:	Two to three parameters: 1. Value (string) from which a substring is to be determined 2. Start position: 1 = First character from left -1 = First character from right 3. Number of characters to be extracted (optional). If this is not specified, a substring running from the specified start position to the end of the value is extracted.

Example:	`<operation name="substring">`
	`<constant>Order processing/Cash sales</constant>`
	`<constant>-1</constant>`
	`<constant>10</constant>`
	`</operation>`

Return value:	**Cash sales** If no value can be determined for one of the parameters or if integers are not specified for the start position and/or number of characters, nothing is returned.

7.1.1.5 if_then_else

Description:	Conditional attribute transformation
Syntax:	Two to three parameters: 1. Value to be checked for equality 2. Value returned in case of equity 3. Value returned in case of inequality

Example:	Value to be checked for equality using the string **true**:
	`<operation name="if_then_else">`
	`<constant>not true</constant>`
	`<constant>Condition is met</constant>`
	`<constant>Condition is not met`
	`</constant>`
	`</operation>`

Return value:	**Condition is not met** If the first value is equal to the string **true** (not case sensitive), the second value is returned, otherwise (if available) the third value. If no value can be determined for the first parameter (condition) or the third parameter if the condition is not met, nothing is returned.

7.1.1.6 set_with_default

Description:	Sets a value to a default value
Syntax:	Two parameters: 1. Value to be returned, if it exists 2. Value to be returned if no value can be determined for the first parameter.

Example:	`<operation name="set_with_default">`
	`<constant>42</constant>`
	`<constant>1</constant>`
	`</operation>`

Return value:	**42** If no value can be determined for both parameters, nothing is returned.

7.1.1.7 integer_plus

Description:	Addition of integer parameters

Syntax:	Two to n parameters: Values to be added

Example:	`<operation name="integer_plus">`
	`  <constant>1</constant>`
	`  <constant>4</constant>`
	`  <constant>37</constant>`
	`</operation>`

Return value:	**42** If no value can be determined for one of the parameters or if one of the parameters has a non-integer value, nothing is returned.

7.1.1.8 integer_minus

Description:	Subtraction of integer parameters from another integer parameter
Syntax:	Two to n parameters: 1. Value from which the other values are to be subtracted 2. Value to be subtracted ... n. Value to be subtracted

Example:	`<operation name="integer_minus">`
	`  <constant>1</constant>`
	`  <constant>4</constant>`
	`  <constant>37</constant>`
	`</operation>`

Return value:	**-40** If no value can be determined for one of the parameters or if one of the parameters has a non-integer value, nothing is returned.

7.1.1.9 string_equal

Description:	Compares strings: Returns the string **true** if all values are equal, otherwise the string **false**.
Syntax:	Two to n parameters: Values to be compared

Example:	`<operation name="string_equal">`
	`  <constant>42</constant>`
	`  <constant>42</constant>`
	`  <constant>37</constant>`
	`</operation>`

Return value:	**false** As soon as a value is found, which does not match the first value checked, **false** is returned even if more values are to follow. If no value can be determined for one of the parameters checked, nothing is returned.

7.1.1.10 string_in

Description:	Compares a string with a list of strings. Returns the string **true** if the string is identical to at least one of the other strings, otherwise the string returned is **false**.
Syntax:	Three to n parameters: Values to be compared

Example:	`<operation name="string_in">`
	`  <constant>42</constant>`
	`  <constant>42</constant>`
	`  <constant>37</constant>`
	`  <constant>42</constant>`
	`</operation>`

Return value:	**true** As soon as a string is found, which is identical to the string to be compared, **true** is returned even if more values are to follow. If no value can be determined for the first parameter, nothing is returned.

7.1.1.11 boolean_not

Description:	Returns the string **false** if the parameter is equal to **true** (not case sensitive), otherwise returns the string **true**.
Syntax:	One parameter: Boolean value to be negated

Example:	`<operation name="boolean_not">`
	`..<constant>this is not true</constant>`
	`</operation>`

Return value:	**true** If no value can be determined for the parameter, nothing is returned.

7.1.1.12 boolean_and

Description:	Returns the string **true** if all parameters are equal to **true** (not case sensitive), otherwise the string **false**.
Syntax:	Two to n parameters: Boolean values to be linked with the Boolean AND

Example:	`<operation name="boolean_and">`
	`..<constant>truE</constant>`
	`..<constant>True</constant>`
	`..<constant>TRUE</constant>`
	`</operation>`

Return value:	**true** As soon as a value is found that is not **true**, **false** is returned. Any subsequent values are not checked. If no value can be determined for one of the parameters checked, nothing is returned.

7.1.1.13 boolean_or

Description:	Returns the string **true** if one of the parameters is equal to **true** (not case sensitive), otherwise returns the string **false**.
Syntax:	Two to n parameters: Boolean values to be linked with the Boolean OR

Example:	`<operation name="boolean_or">`
	`  <constant>false</constant>`
	`  <constant>True</constant>`
	`  <constant>this is not true</constant>`
	`</operation>`

Return value:	**true** As soon as a value is found that is **true**, **true** is returned. Any subsequent values are not checked. If no value can be determined for one of the parameters checked, nothing is returned.

7.1.1.14 exists

Description:	Existence check
Syntax:	One to n parameters: Values whose existence is to be checked

Example:	`<operation name="exists">`
	`  <constant>false</constant>`
	`  <constant>Order volume</constant>`
	`  <constant>Turnover</constant>`
	`</operation>`

Return value:	**true** Returns **true** if a value could be retrieved for all parameters, otherwise **false**.

7.1.1.15 get_null

Description:	Returns nothing. If the outermost operation in a transformation rule returns nothing, the attribute type is not transferred to the XML output file.
Syntax:	No parameters
Example:	`<operation name="get_null"/>`
Return value:	None

7.1.1.16 getCurrentTimestamp

Description:	Returns a current time stamp. The format is specified in the first parameter. It includes all date and time formats specified by the **java.text.SimpleDateFormat** class (like the **creationtimestamp** condition operator), as described in the **Data import** technical reference in the **Transformations** section. If you only specify one parameter, the same time stamp - corresponding to the first time the operation is called - will be generated as the attribute value in all system events in the output file(s). If you specify two parameters, the time of the operation call is written to the system event as a time stamp attribute. This means the values can differ depending on the selected format and extraction duration. There is no plausibility check of the second parameter, it just needs to exist. If no value can be determined for the first parameter, nothing is returned.
Syntax:	One to two parameters

Example:	`<operation name="getCurrentTimestamp">`
	`  <constant>dd.MM.yyyy HH:mm:ss</constant>`
	`  <constant/>`
	`</operation>`

Return value:	for example, **20.07.2006 15:45:08**, otherwise various values for different system events depending on extraction duration.

7.1.2 Example of attribute transformation

Importing source system data has generated the following entry in the XML output file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

You want to link the value of the **VBAP-ERNAM**attribute type with a constant string and write it to the new **Order item recorded by** attribute type. In addition, you want to create the **Note on order recording** attribute type, which extracts the first name and last name from the **VBAP-ERNAM** attribute type and creates the following string:

<First name>< ><Last name>< has recorded the item ><VBAP-POSNR>< of the order ><VBAP-VBELN>< .>

The information in angle brackets represents attribute values that are extracted or transformed from existing attribute types or constants. You can specify the necessary information in the XML attribute transformation file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE attributecalculation SYSTEM
                        "attributetransformation.dtd">
<attributecalculation>
  <calculateattribute type="Order item recorded by">
    <operation name="concat">
      <constant>Walter, "A" </constant>
      <valueof>VBAP-ERNAM</valueof>
    </operation>
  </calculateattribute>
  <calculateattribute type="Note on order recording">
    <operation name="concat">
      <operation name="substring">
        <valueof>Order item recorded by</valueof>
        <constant>1</constant>
        <constant>6</constant>
      </operation>
      <constant> </constant>
      <operation name="substring">
        <valueof>Order item recorded by</valueof>
```

```
      <constant>-1</constant>
      <constant>7</constant>
    </operation>
    <constant> recorded item</constant>
    <valueof>VBAP-POSNR</valueof>
    <constant> in order </constant>
    <valueof>VBAP-VBELN</valueof>
    <constant>.</constant>
  </operation>
 </calculateattribute>
</attributecalculation>
```

When you call up the relevant command line program again, this time with the **-calcconfig <XML attribute transformation file>** parameter, the following XML output file is generated:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
 <event>
  <attribute type="VBAP-VBELN">3866</attribute>
  <attribute type="VBAP-POSNR">20</attribute>
  <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
  <attribute type="Order item recorded by">
              Walter, "A" SCHMIDT
  </attribute>
  <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
  <attribute type="VBAP-KWMENG">3</attribute>
  <attribute type="Note on order recording">
  Walter SCHMIDT recorded item 20 in order 3866.
  </attribute>
 </event>
</eventlist>
```

7.2 Data source

All configuration files required for extraction and the output file can be specified together in a data source file. The XML file is specified by the **-datasource** parameter on the console. This parameter is used instead of the **-systemconfig**, **-tableconfig**, **-csvconfig**, **-i**, **-calcconfig**, **-outfile** and **-nozip** parameters.

The format of the XML file is specified by the **datasource.dtd** DTD:

XML element	XML attribute	Description
datasource	name type lastreaddate lastreadtime	A data source has a unique name, is of exactly one of the **EVENT, GRAPH, MYSAP, JDBC** or **CSV** types and contains the last time of extraction.
dataextraction (see chapter		Specifies the XML output file(s). The file extension (**.zip** or **.xml**)

XML element	XML attribute	Description
Configuration of multiple output files (page 160))		must be explicitly specified. Either a zipped or unzipped XML file is generated depending on the file extension.
systemconfig		Specifies the system configuration file. The file may only contain one system access configuration.
eventspec		Specifies the table configuration file (event specification). The file may contain only one table configuration (**configuration** XML element).
attribute transformation (optional)		Specifies the attribute transformation file

If you transfer the data source file to the command line program with the **-datasource** parameter, the last time of extraction (**lastreaddate**, **lastreadtime**) is used and updated only in this file. The time in the system configuration used is no longer adjusted.

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="BILLING" type="MYSAP"
      lastreaddate="19700101" lastreadtime="000000">
   <description name="desc_de" language="de">
                              (Cancelation) invoice,
                            (Cancelation) credit memo,
                             Internal credit memo,
                            Internal allocation,
                     Debit memo, pro forma invoice
   </description>
   <description name="desc_en" language="en">
                          (Cancelation) invoice,
                      (Cancelation) credit memo,
                           Internal credit memo,
                            Internal allocation,
                     Debit memo, pro forma invoice
   </description>
   <dataextraction> see chapter
      Configuration of multiple output files (page 160)
   </dataextraction>
   <systemconfig>C:\Programs\jdbc2ppm\xml\Systemconfig.xml
   </systemconfig>
```

```
   <eventspec>C:\Programs\jdbc2ppm\xml\TableConfiguration.xml
   </eventspec>
   <attributetransformation>C:\Programs\jdbc2ppm\xml\
                        attributecalculation.xml
   </attributetransformation>
</datasource>
```

7.2.1 Configuration of multiple output files

If very large data volumes are extracted and written to a single output file as system events, the file may become difficult to handle. In such cases, it is possible to configure the data source used in such a way that the extracted data is written to any number of XML output files. All you need to do is specify the maximum number of system events per output file and the output file name in the **dataextraction** XML element.

Example (for a CSV data source, same procedure for JDBC and SAP types)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="BILLING" type="CSV">
  <dataextraction>
    <outputfilename>..\custom\<clientname>\data\
      BILLING_data_$EXTRACTIONDATE$_$EXTRACTIONTIME$.zip
    </outputfilename>
    <numberofeventsperxmlfile>
      100000
    </numberofeventsperxmlfile>
  </dataextraction>
  ...
  <systemconfig>...</systemconfig>
  <eventspec>...</eventspec>
  ...
</datasource>
```

For the **BILLING** data source, the **numberofeventsperxmlfile** XML element specifies that each XML output file is to contain 100000 system events, with the last output file generated containing all remaining events.

The **outputfilename** XML element specifies the path and name of the output files. These settings generate XML files with names in the form **BILLING_data_$EXTRACTIONDATE$_$EXTRACTIONTIME$.zip** in the specified output directory. The **$EXTRACTIONDATE$** name variable contains the extraction date, while **$EXTRACTIONTIME$** contains the extraction time. The output files generated are numbered by placing **_<x>** at the end of the name, where **x** is a consecutive number and the first output file generated is not numbered.

If 369000 system events were extracted on 23 June 2007 at 11:36:45 using the example configuration shown, the following output files would be created:

- **BILLING_data_20070623_113645.zip**
  (with 100000 system events)
- **BILLING_data_20070623_113645_1.zip**
  (with 100000 system events)
- **BILLING_data_20070623_113645_2.zip**
  (with 100000 system events)
- **BILLING_data_20070623_113645_3.zip**
  (with 69000 system events)

The table below lists all configuration options:

XML element	Description
numberofeventsperxmlfile	Fixed number of system events written to an output file. The last output file created contains the remaining number of events. If this element is missing, all system events are written to one output file.
outputfilename	Path and naming pattern of output files. XML and ZIP formats are supported. A ZIP output file contains an XML output file with the same name. The output files for an extraction are consecutively numbered by **_<x>** at the end of the name, although the first file created is not numbered.

The following variables are permitted in the output file name (**outputfilename**) and can be used in any combination:

Variable (data source type)	Description
$EXTRACTIONDATE$ (CSV, SAP, JDBC)	Extraction date (format: yyyyMMdd)
$EXTRACTIONTIME$ (CSV, SAP, JDBC)	Extraction time (format: HHmmss)
$BEGINDATE$ (SAP, JDBC)	Start date of extract period (format: yyyyMMdd)
$BEGINTIME$ (SAP, JDBC)	Start time of extract period (format: HHmmss)

Variable (data source type)	Description
$ENDDATE$ (SAP, JDBC)	End date of extract period (format: yyyyMMdd)
$ENDTIME$ (SAP, JDBC)	End time of extract period (format: HHmmss)
$VALUECONSTRAINT$ (SAP, see chapter Condition operators (page 26) and JDBC, see chapter Condition operators (page 89))	Output format: <Operator>_Value1 or <Operator>_Value1_<Operator>_Value2 Output of operators and integer comparison values used to restrict the volume of data extracted. The name of the output file could be: **outfile_gt_230_le_300_<x>.xml**, where **x** is the consecutive number. Operator representation: greater than: **gt** greater than or equal to: **ge** less than: **lt** less than or equal to: **le**

You can conveniently set the distribution of the extracted data sets to multiple output files in PPM Customizing Toolkit in the **Data source management** of the client, via **Additional settings...** in the **Data extraction** area.

7.2.2 Configuration of an offset extraction in terms of time or value

The functions described in this chapter are supported only by the two process extractors **PPM Process Extractor JDBC-2-PPM** and **PPM Process Extractor SAP-2-PPM** and only when using a data source configuration.

For the time-based extraction, you can specify a time value in the data source configuration. The value is subtracted from the current execution time as the end time when executing the corresponding command line program without end time parameters (**-enddate** or **-endtime**). Data extraction will end earlier by the specified value (in seconds).

Without specifying a time offset value, data records that have been created but not written explicitly to the database to be extracted might not be extracted in any of two subsequent extraction operations due to their offset time stamp (see **Continuous automated extraction** (page 68)). This can be prevented by specifying an offset value.

Example (SAP data source)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">

<datasource name="SAP" type="MYSAP" lastreaddate="19700101"
                lastreadtime="000000" readoffset="86400">
    <description name="default_description" language="en"/>
    <description name="default_description" language="de">
       SAP
    </description>
    ...
</datasource>
```

The **readoffset** XML attribute indicates an offset value of **86400 seconds**. When calling the command line program with

```
runjdbc2ppm -datasource SAP.xml
```

the extraction operation ends one day (86400 seconds) before the current execution time. If the program execution time is 11.08.08 9:48:05, the extraction operation ends on 10.08.08 at 9:48:05.

```
...
I: 11.08.08 09:48:06: [XML] Reading data with start date/time 19700101/000000
and end date/time 20080810/094805...
...
```

The offset value of the last extraction time is written to the data source file as the start time for the next data extraction.

For the conditional extraction using an integer criterion (logical operator **valueconstraint**, see **Extraction using conditions** (page 87)), you can similarly specify an offset value that is subtracted from the start value (**lastreadvalue** from the data source file) when running the program. If the result of your data is a negative value for the integer criterion, this value is handled like a positive value.

An offset value specified in the data source configuration is included in data extraction only if the end time or the lower limit of the integer criterion is not explicitly specified.

Tip

You can easily configure offset values in **PPM Customizing Toolkit** in the **Client** module group under **Data source management**. Enter the value as additional information for the last extraction time or the last value extracted in the components of the relevant data source.

SPECIFY A TIME ZONE

If you have not specified the end time in the command line (**-enddate** or **-endtime** arguments) the current time of the local computer is used as end time by default. This end

time is used for the **creationtimestamp**, **date_creationtimestamp**, or **char_creationtimestamp** operators to restrict the data volume to be extracted. If the data field referenced in the operators contains time stamps assigned to a different time zone, you can specify this time zone in the **sourcefieldtimezone** XML attribute (for example, **Australia/Canberra**). In the subsequent data extraction, the local computer time is converted to the time zone specified, and after successful extraction the time is saved as last time of extraction (**lastreaddate** and **lastreadtime** XML attributes) in the data source.

If no valid time zone or the attribute itself is not specified in the **sourcefieldtimezone** XML attribute, a time zone conversion of the end time will not be carried out. With CTK, the time zone can be easily selected using the interface.

Time zone examples

Africa/Dakar, Africa/Johannesburg, Australia/Sydney, America/Denver, America/Los_Angeles, America/Mexico_City, Canada/Central, Europe/Berlin, Europe/London, Mexico/General, US/Central, US/Hawaii

7.3 XML output file (formats)

By default, data extracted using the PPM process extractors is written to an XML output file in PPM system event format.
By specifying one of the optional command line parameters **-pikidatamapping** or **-dimdatamapping**, you can also have the extracted data written to the output file(s) in process instance-independent measure data format or dimension data format.
The different output formats are described in detail in the **PPM Data import** manual.

7.3.1 PPM system event format

Default format of the XML output file(s) if neither of the **-pikidatamapping** and **-dimdatamapping** command line parameters is used (see chapter **XML output file (PPM system event format)** (page 35)).

7.3.2 PIKIDATA format

The data extracted is to be written as process instance-independent measures (PIKI) to an XML output file for the PPM import using the **runpikidata.bat** command line program.

The XML configuration file has the following basic structure:

. . .

```
<pikidatamapping>
  <pikicube name="...">
    <pikicolmapping>
      <datacol name="..."/>
      <keyspec key="..."/>
    </pikicolmapping>
    ...
  </pikicube>
  ...
</pikidatamapping>
```

XML element	XML attribute	Description
pikidatamapping		Root element of mapping definition. Contains a list of PPM measure series to be configured.
pikicube	name	Identifier of PPM measure series (corresponds to the value of the **name** XML attribute for the measure series configuration in PPM)
pikicolmapping		Groups configuration elements for a data column
datacol	name	Name of the data column in which the PIKI values are saved (corresponds to the internal name of the PIKI in the measure configuration)
	format (optional)	Format string for transformation of data value
keyspec	key	Name of the key for the key-value assignment
	desckey	Name of the key for the key-description assignment
value		Value of the process instance-independent measure

The file extract below shows the existing **PCA Import** PPM measure series:

```
...
<pikicube name="PIKICUBE_TURNOVER">
  <description language="de" name="PCA-Import"/>
```

```
  <description language="en" name="PCA Import"/>
  <pikidef name="TURNOVER" retrievertype="NUM_KEYINDICATOR"
           dimreferring="LOOSE" kigroup="KI_GROUP_COST">
    <description language="de" name="Umsatz">
                               Umsatz
    </description>
    <description language="en" name="Sales revenues">
                               Sales revenues
    </description>
    <datatype name="COST"/>
  </pikidef>
  <refdim name="VKORG"/>
  <refdim name="DIVISION"/>
  <refdim name="TIME" refinement="BY_MONTH"/>
</pikicube>
...
```

Example

The following key-value pairs are the expected result of the extraction in the XML output file in PPM system event format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="MARA-TURNOVER">225489 EUR</attribute>
    <attribute type="MARB-VKORG_ID">1000</attribute>
    <attribute type="MARB-VKORG_DESC">Germany Hamburg
    </attribute>
    <attribute type="DIM_ROUGH_ID">01</attribute>
    <attribute type="DIM_ROUGH_DESC">Product category 01
    </attribute>
    <attribute type="DIM_DETAILED_ID">8112</attribute>
    <attribute type="DIM_DETAILED_DESC">8112 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
  <event>
    <attribute type="MARA-TURNOVER">135699 EUR</attribute>
    <attribute type="MARB-VKORG_ID">4000</attribute>
    <attribute type="MARB-VKORG_DESC">Austria Vienna
    </attribute>
    <attribute type="DIM_ROUGH_ID">01</attribute>
    <attribute type="DIM_ROUGH_DESC">Product category 01
    </attribute>
    <attribute type="DIM_DETAILED_ID">8112</attribute>
    <attribute type="DIM_DETAILED_DESC">8112 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
  <event>
    <attribute type="MARA-TURNOVER">363521 EUR</attribute>
    <attribute type="MARB-VKORG_ID">1000</attribute>
    <attribute type="MARB-VKORG_DESC">Germany Hamburg
    </attribute>
    <attribute type="DIM_ROUGH_ID">07</attribute>
```

```
    <attribute type="DIM_ROUGH_DESC">High Tech</attribute>
    <attribute type="DIM_DETAILED_ID">9128</attribute>
    <attribute type="DIM_DETAILED_DESC">9128 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
</eventlist>
```

The mapping file below transfers the key-value pairs into the PPM-compatible XML format for importing PIKI values:

```
...
<pikidatamapping>
  <pikicube name="PIKICUBE_TURNOVER">
    <pikicolmapping>
      <datacol name="TURNOVER"/>
      <keyspec key="MARA-TURNOVER"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="VKORG"/>
      <keyspec key="MARB-VKORG_ID"
                          desckey="MARB-VKORG_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="DIVISION"/>
      <keyspec key="DIM_ROUGH_ID"
                           desckey="DIM_ROUGH_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="DIVISION"/>
      <keyspec key="DIM_DETAILED_ID"
                         desckey="DIM_DETAILED_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="TIME" format="MMM yyyy"/>
      <keyspec key="CAL-TIME"/>
    </pikicolmapping>
  </pikicube>
</pikidatamapping>
```

Using this mapping file generates the following XML output file when you run the program:

```
...
<pikidata>
  <pikicube name="PIKICUBE_TURNOVER">
    <datacols>
      <datacol name="TURNOVER"/>
      <datacol name="VKORG"/>
      <datacol name="DIVISION"/>
      <datacol name="DIVISION"/>
      <datacol name="TIME" format="MMM yyyy"/>
    </datacols>
    ...
    <datarow>
      <value>225489 EUR</value>
      <value>1000{Germany Hamburg}</value>
      <value>01{Product category 01}</value>
      <value>8112{8112 description}</value>
```

```
      <value>January 2002</value>
    </datarow>
    <datarow>
      <value>135699 EUR</value>
      <value>4000{Austria Vienna}</value>
      <value>01{Product category 01}</value>
      <value>8112{8112 description}</value>
      <value>January 2002</value>
    </datarow>
    <datarow>
      <value>363521 EUR</value>
      <value>1000{Germany Hamburg}</value>
      <value>07{High Tech}</value>
      <value>9128{9128 description}</value>
      <value>January 2002</value>
    </datarow>
    ...
  </pikicube>
</pikidata>
...
```

7.3.3 DIMDATA format

The data extracted by the relevant PPM process extractor is to be written to the XML output file in the special import format for one, two or multi-level dimensions. XML output files in this format can be imported into PPM using the **rundimdata.bat** command line program (see **PPM Data import** technical reference).

To extract data correctly, the relevant process extractor requires a mapping file that assigns the keys (and any descriptions) for the individual dimension levels to the correct extracted field values. This XML configuration file has the following structure:

```
...
<dimdatamapping>
  <dimension name="...">
    <dimcolmapping>
      <datacol name="..."/>
      <keyspec key="..."/>
    </dimcolmapping>
    ...
  </dimension>
</dimdatamapping>
...
```

XML element	XML attribute	Description
dimdatamapping		Root element of mapping definition. Contains a list of PPM dimensions to be configured.

XML element	XML attribute	Description
dimension	name	Identifier of the PPM dimension (corresponds to the value of the **name** XML attribute from the dimension configuration in the PPM system)
dimcolmapping		Groups configuration elements for a data column.
datacol	name	Name of the data column, in which the dimension values are saved.
	format (optional)	Format string for transformation of the data value for use in the PPM system
keyspec	key	Name of the key for the extracted key-value assignments
	desckey	Name of the key for the extracted key-description assignments
value		Specification of a constant dimension value

Example

The following key-value pairs are the expected result of the extraction in the XML output file in PPM system event format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="MAT1_ID-MATNR">1000</attribute>
    <attribute type="MAT1_DESC-HTEXT">Motor44</attribute>
    <attribute type="MAT2_ID-MTART">32</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Components</attribute>
  </event>
  <event>
    <attribute type="MAT1_ID-MATNR">1001</attribute>
    <attribute type="MAT1_DESC-HTEXT">Wrench77</attribute>
    <attribute type="MAT2_ID-MTART">45</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Tools</attribute>
  </event>
  <event>
    <attribute type="MAT1_ID-MATNR">1002</attribute>
```

```
    <attribute type="MAT1_DESC-HTEXT">Pump43</attribute>
    <attribute type="MAT2_ID-MTART">33</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Trade goods</attribute>
  </event>
</eventlist>
```

The mapping file below transfers the key-value pairs into the PPM-compatible XML format for importing dimension values:

```
...
<dimdatamapping>
  <dimension name="MATERIAL">
    <dimcolmapping>
      <datacol name="LEVEL1_ID"/>
      <keyspec key="MAT1_ID-MATNR"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL1_DESC"/>
      <keyspec key="MAT1_DESC-HTEXT"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL2_ID"/>
      <keyspec key="MAT2_ID-MTART"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL2_DESC"
      <keyspec key="MAT2_DESC-MTBEZ"/>
    </dimcolmapping>
  </dimension>
</dimdatamapping>
```

The names of the data columns for a PPM dimension level are fixed (**LEVEL1_ID**, **LEVEL1_DESC**, **LEVEL2_ID**, **LEVEL2_DESC**, etc.) and must not be changed.

The table below illustrates the assignment of the data columns to the keys and descriptions of the individual dimension levels:

dimdata configuration	Dimension configuration
LEVEL1_ID	Identifier of first dimension level
LEVEL1_DESC	Description of first dimension level
LEVEL2_ID	Identifier of second dimension level
LEVEL2_DESC	Description of second dimension level
LEVELn_ID	ID of the n-th dimension level
LEVELn_DESC	Description of the n-th dimension level

The XML output file generated in DIMDATA format has the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dimdata SYSTEM "dimdata.dtd">
<dimdata>
  <dim name="MATERIAL">
    <datacols>
      <datacol name="LEVEL1_ID"/>
      <datacol name="LEVEL1_DESC"/>
      <datacol name="LEVEL2_ID"/>
      <datacol name="LEVEL2_DESC"/>
    </datacols>
    <datarow>
      <value>1001</value>
      <value>Motor44</value>
      <value>32</value>
      <value>Components</value>
    </datarow>
    <datarow>
      <value>1002</value>
      <value>Wrench77</value>
      <value>45</value>
      <value>Tools</value>
      </datarow>
    <datarow>
      <value>1003</value>
      <value>Pump43</value>
      <value>33</value>
      <value>Trade goods</value>
    </datarow>
  </dim>
```

8 Data pseudonymization

You can pseudonymize the data extracted by the **SAP**, **JDBC**, and **CSV**, extractors. The extracted data is pseudonymized before it is written to the output file as a source system event.

You can pseudonymize the source data using CTK. The **Use pseudonymization during data extraction** option is disabled by default but you can enable the option in the client properties in CTK. If you enable this option, CTK provides the **Pseudonymization** tab in the **Data extraction** component in the **Process merge** module. On this page, you can specify for each available source system attribute whether its value is to be pseudonymized or not.

Open a client in CTK -> Client properties -> Data sources -> Enable **Use pseudonymization during data extraction**.

For the data pseudonymization, CTK creates a required key and stores it in the **pseudonymization.key** file in the **.../custom/<client>/keyfiles** directory. The created **pseudonymization.key** file is of binary type.

Note that you must enable pseudonymization before any data for a source system attribute is extracted. Already imported cleartext data cannot be pseudonymized.

Warning

We recommend that you create a backup of the **pseudonymization.key** file. The key file cannot be restored and you cannot use another pseudonymization key instead.

8.1 Scaled systems

If pseudonymization is to be used for scaled systems, all sub-servers must use the same pseudonymization key, that is, you must ensure that the same key file exists on all sub-servers.

Therefore, if pseudonymization is activated on sub-server or master, copy the key file from this PPM client to all other sub-servers and to the master of this scaled scenario in the **.../custom/<client>/keyfiles** directory. This must be done before the first data extraction of pseudonymized attributes on the respective client.

8.2 Configuration of pseudonymized attributes

The **datasource.dtd** DTD is adapted for pseudonymization as follows:

```
<!ELEMENT datasource (description*, realmtable?, dataextraction?, data,
archive?, fragments?, mapping?, systemconfig?, tables?, eventspec?,
attributetransformation?, eventattributetypes?, attributesettings?,
edaeventtype?, pseudonymization?)>
```

```
<!ELEMENT eventattribute (#PCDATA)>
<!ELEMENT pseudonymization (eventattribute+)>
```

The optional element **pseudonymization** must contain at least one **eventattribute** sub-element. A **eventattribute** element contains the name of the Source System attribute to be pseudonymized, expressed as the attribute type that appears in the event in the output file.

Example

The following example shows the pseudonymization of BSEG-ERNAME and LAST_USER.

- Output file without pseudonymization

```
<event>
              <attribute type="BSEG-ERNAME">Test User</attribute>
              <attribute type="CARDNO">308342023837750</attribute>
              <attribute type="CONTACT_REASON">Bonus not
received</attribute>
              <attribute type="FIELD_2">Phone</attribute>
              <attribute type="LAST_USER">Second user</attribute>
              <attribute type="SR_CLOSED_DATE">11.06.2003
18:07:52</attribute>
</event>
```

- Configuration of pseudonymization

```
<pseudonymization>
            <eventattribute>BSEG-ERNAME</eventattribute>
            <eventattribute>LAST_USER</eventattribute>
</pseudonymization>
```

- Output file with pseudonymization

```
<event>
              <attribute
type="BSEG-ERNAME">WReKoRCNHVdJF6FZ8oLqFGg==</attribute>
              <attribute type="CARDNO">308342023837750</attribute>
              <attribute type="CONTACT_REASON">Bonus not
received</attribute>
              <attribute type="FIELD_2">Phone</attribute>
              <attribute
type="LAST_USER">+XMLp1ip51d++gfVWvA+bQ==</attribute>
              <attribute type="SR_CLOSED_DATE">11.06.2003
18:07:52</attribute>
</event>
```

8.3 Dimensions of pseudonymized data

All pseudonymized data are of data type TEXT. This means the data must have the data type TEXT in PPM, including data that was, for example, of data type NUMERICAL before pseudonymization.

Therefore, you must ensure that the values are assigned to PPM text attributes during XML import and are only used for text dimensions.

The length of the pseudonymized value (cipherLength) depends on the length of the value before pseudonymization (clearLength).

You must create dimensions in which pseudonymized data are to be stored with a dimension length of 1000. Unencrypted plain text data must not exceed 200 characters, otherwise the pseudonymization algorithm may generate values longer than 1000. This would make conversion back into plain text impossible.

The PPM import logs an error message for every dimension value which is too long.

8.4 PPM pseudonymization tool

The PPM pseudonymization tool pseudonymizes a clear text value or converts a pseudonymized value to clear text value.

You must add the credentials of a PPM user with system administration rights to the following command line examples:

- "runppmpseudonymize -in <file> [-encoding <encoding>]..." generates a pseudonymized value.
- "runppmpseudonymize -reverse <pseudonymized value> -out <file> [-encoding <encoding>]..." generates a cleartext value.

```
runppmpseudonymize -user <username> -password <password> [-client <name>]
    {-in <filename>} | {-reverse <pseudonymized value> -out <filename>}
    [-encoding <encoding>]
    [-version]
    [-language <ISO code>][protocoloptions]
```

Parameter	Description
-in <filename>	Name of the file which contains the cleartext value which should be pseudonymized
-reverse <pseudonymized value>	If this parameter is specified, then this pseudonymized value will be converted to cleartext value.
-out <filename>	Name of the file to which the cleartext value should be written.

Parameter	Description
-encoding <encoding>	Encoding of the in- or out-file (Default: UTF-8)
-language <ISO code>	Language
-version	Version number of the application and the database schema
protocoloptions	protocoloptions can consist of the following instructions: -protocolfile <filename> Logging to file <file name> -information {yes|no|default} Logging of information -warning {yes|no|default} Logging of warnings -error {yes|no|default} Logging of errors

If the **-reverse <pseudonymized value>** parameter is specified, the plain text value is output to the file **-out <filename>** in the specified encoding. If the file **filename** already exists, it will be overwritten.
If the **-in <filename>** parameter is specified, the value in the file **<filename>** is interpreted as a plain text value and the pseudonymized value is output directly on the standard output, that is, not in the protocol. The file is read in the specified encoding.

The key file **pseudonymization.key** used for encryption/decryption is expected in the following directory: **<PPM installation>\server\bin\work\data_ppm\custom\<client>\keyfiles**. Therefore, the client must also be specified in the command line using the **-client** parameter. If the specified client does not support pseudonymization, a corresponding error message appears.

Note that the user used for (de-) pseudonymization with this tool must have system administration rights in PPM.

8.5 Adding new encrypted attributes

In the **Data import** module of CTK, you must set the TEXT data type for the attribute before the first mapping so that the correct data type is assigned to the PPM attributes automatically created during mapping.

If you specified in the data extraction configuration for the source system event attribute "VDARL-RERF", which actually has the data type DOUBLE in source system, that it is to be encrypted, you must now select the data type TEXT, since the values of encrypted source system event attributes are texts.

- The maximum length of 200 of clear text value of source system attribute must not be exceeded.

- If a pseudonymized value is to be used as a dimension value, you must ensure that the dimension length is set to 1000.
- Also for integer and double values, the maximum length of the values must not be exceeded.

9 Legal information

9.1 Documentation scope

The information provided describes the settings and features as they were at the time of publishing. Since documentation and software are subject to different production cycles, the description of settings and features may differ from actual settings and features. Information about discrepancies is provided in the Release Notes that accompany the product. Please read the Release Notes and take the information into account when installing, setting up, and using the product.

If you want to install technical and/or business system functions without using the consulting services provided by Software AG, you require extensive knowledge of the system to be installed, its intended purpose, the target systems, and their various dependencies. Due to the number of platforms and interdependent hardware and software configurations, we can describe only specific installations. It is not possible to document all settings and dependencies.

When you combine various technologies, please observe the manufacturers' instructions, particularly announcements concerning releases on their Internet pages. We cannot guarantee proper functioning and installation of approved third-party systems and do not support them. Always follow the instructions provided in the installation manuals of the relevant manufacturers. If you experience difficulties, please contact the relevant manufacturer.

If you need help installing third-party systems, contact your local Software AG sales organization. Please note that this type of manufacturer-specific or customer-specific customization is not covered by the standard Software AG software maintenance agreement and can be performed only on special request and agreement.

9.2 Support

If you have any questions on specific installations that you cannot perform yourself, contact your local Software AG sales organization (https://www.softwareag.com/corporate/company/global/offices/default.html). To get detailed information and support, use our websites.

If you have a valid support contract, you can contact **Global Support ARIS** at: **+800 ARISHELP**. If this number is not supported by your telephone provider, please refer to our Global Support Contact Directory.

ARIS COMMUNITY

Find information, expert articles, issue resolution, videos, and communication with other ARIS users. If you do not yet have an account, register at ARIS Community**.**

SOFTWARE AG EMPOWER PORTAL

You can find documentation on the Software AG Documentation website (https://empower.softwareag.com/). The site requires credentials for Software AG's Product Support site **Empower**. If you do not yet have an account for **Empower**, send an e-mail to empower@softwareag.com with your name, company, and company e-mail address and request an account.

If you have no account, you can use numerous links on the TECHcommunity website. For any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory and give us a call.

TECHCOMMUNITY

On the **TECHcommunity** website, you can find documentation and other technical information:

- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Access articles, code samples, demos, and tutorials.
- Find links to external websites that discuss open standards and web technology.
- Access product documentation, if you have **TECHcommunity** credentials. If you do not, you will need to register and specify **Documentation** as an area of interest.

EMPOWER (LOGIN REQUIRED)

If you have an account for **Empower**, use the following sites to find detailed information or get support:

- You can find product information on the Software AG Empower Product Support website.
- To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the Knowledge Center.
- Once you have an account, you can open Support Incidents online via the eService section of Empower.
- To submit feature/enhancement requests, get information about product availability, and download products, go to Products.

SOFTWARE AG MANAGED LEARNINGS

Get more information and trainings to learn from your laptop computer, tablet or smartphone. Get the knowledge you need to succeed and make each and every project a success with expert training from Software AG.

If you do not have an account, register as a customer or as a partner.