

ITP

# ITP and XML

2001-12-28

*This white paper describes using ITP with XML data to create advanced  
business document production solutions.*



**INTELLIGENT TEXT PROCESSING**



# Table of contents

- Table of contents ..... 1
- 1. Introduction..... 2
- 2. Introduction to ITP..... 3
  - 2.1 The architecture of ITP .....3
  - 2.2 Database abstraction .....4
  - 2.3 Model documents, models and result documents.....4
  - 2.4 The ITP document development process .....4
  - 2.5 Producing result documents.....5
  - 2.6 Printing ITP output .....5
  - 2.7 The ITP product family.....5
- 3. The ITP Instruction language ..... 7
- 4. XML..... 11
  - 4.1 A very short introduction to XML ..... 11
  - 4.2 Data exchange with XML..... 11
  - 4.3 ITP and XML ..... 12
- 5. The ITP XML File Connection ..... 13
  - 5.1 ITP Concepts and XML ..... 13
    - 5.1.1 Fields and entries .....13
    - 5.1.2 Subentries.....13
    - 5.1.3 Plural and singular entries and optional elements.....13
    - 5.1.4 The XML input file .....13
  - 5.2 The DID..... 14
    - 5.2.1 Entries .....14
    - 5.2.2 Fields.....14
  - 5.3 Examples ..... 14
  - 5.4 The XML File Connection Wizard..... 15
  - 5.5 Conformance to XML standards..... 15
    - 5.5.1 Character encodings.....15
- 6. Conclusion..... 16



# 1. Introduction

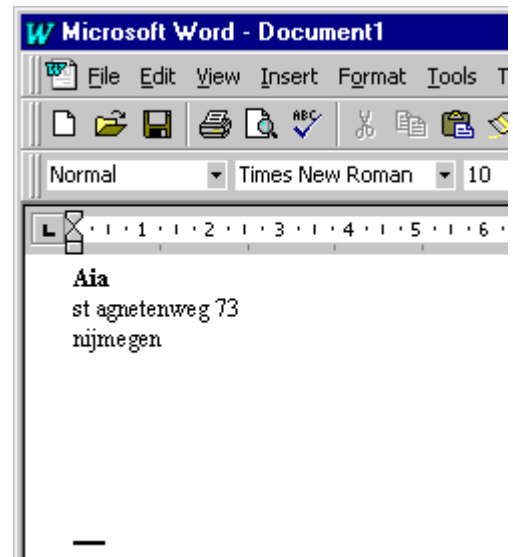
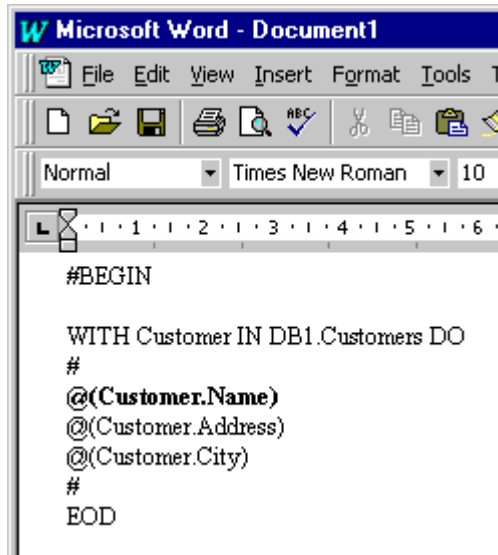
This white paper describes the ITP product and using XML data with ITP.

ITP is a client/server software solution that offers unprecedented flexibility in creating text/data merge applications for modern word processors, like Microsoft Word and Corel WordPerfect. ITP connects these word processors to your databases. It does so by adding a flexible and easy-to-use instruction language to your word processor of choice.

ITP supports access to multiple database types. These types of access are called ITP Connections. Currently ITP (version 2.1) supports access to Lotus Notes/Domino databases, AS/400 database, Oracle databases, ODBC enabled databases and XML data.

This white paper focuses on ITP's access to XML data. We start with an introduction to ITP and to ITP document development. We then give a short introduction to XML and finally we discuss XML data access for ITP.

White paper



## 2. Introduction to ITP

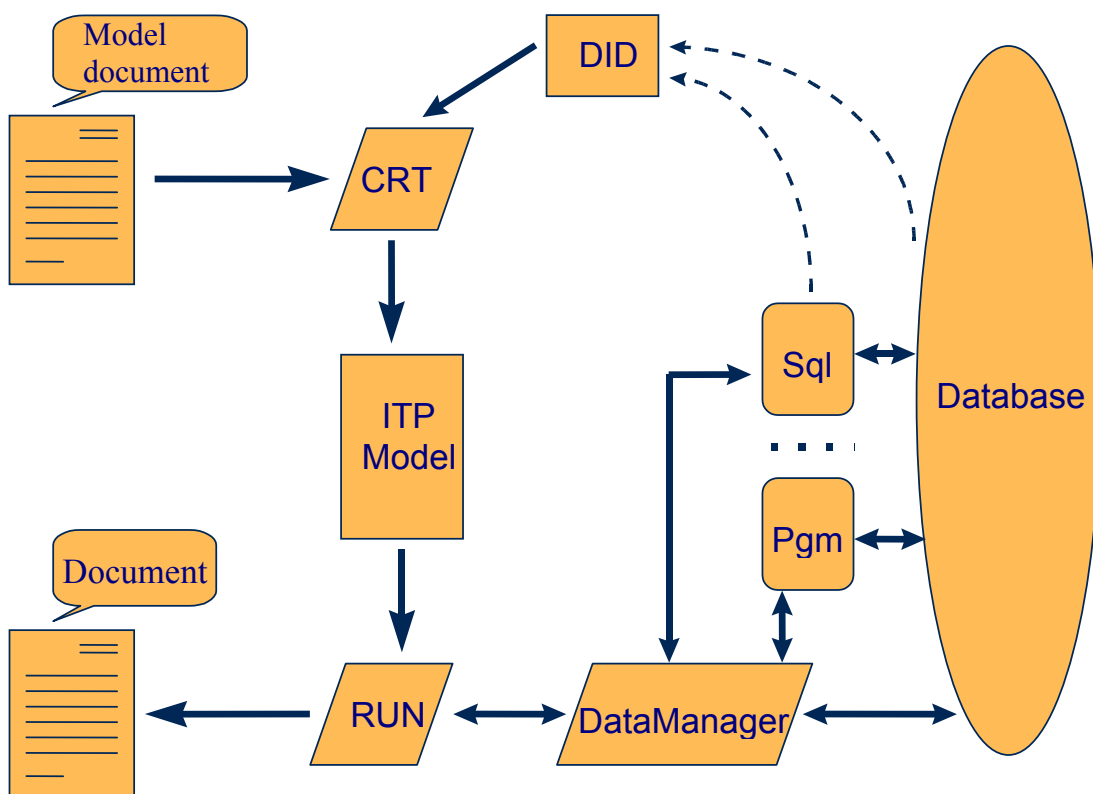
With ITP the model documents (merge documents) are developed in the word processor of your choice. After development of a model document, ITP creates the final output document(s) based on this model document. The word processor itself is not used during this creation process.

ITP combines all familiar word processor facilities, both textual as well as layout, in a seamless way with data from your corporate databases. Text and layout in the documents can be conditioned based on the data from the database. ITP accesses the database in real-time, so no downloading, pre-processing or copying of data is required.

Document production can be document driven, where the output is based on conditions and calculations coded in your document that may or may not use database access. It can also be data driven, where the output is based on the contents of the database. If necessary, the document production or parts of it can be controlled by user interaction.

Very different types of documents can be developed with ITP, ranging from correspondence, invoices and quotations to policies, contracts and financial reports.

### 2.1 The architecture of ITP



This figure describes all basic ITP processes:

1. A description of the database is created, the DID. The DIDs describes how to access the database using for example SQL queries or external data retrieval programs.
2. Model documents are developed in the word processor of your choice. These documents contain ITP instructions with references to the ITP database description. Model documents are compiled to ITP Models.



3. ITP Models are invoked to produce the actual result documents. During this process data are accessed using the SQL queries, programs etc. which were described in the DID. This step is the actual production process.

## 2.2 Database abstraction

ITP uses its own database descriptions for database access. In ITP language, these are called DIDs: **Database Interface Definition**. These database descriptions can contain multiple definitions of sets of related data (e.g. a database record) and the information required to retrieve this data. Relations between these definitions can be defined as well.

Through these definitions ITP models can access data without any knowledge of where the data resides or how to access it.

The DID is in fact an abstract view on a, possibly imaginary, database that maps to database files, external data retrieval programs or SQL queries. This mapping can be one-on-one, meaning that the abstract view closely mimics the original database, but it can also be completely different, offering a much more abstract (high-level) view of the database, which simplifies the development of model documents, especially when dealing with highly normalised databases.

By mapping the abstract view on the database to external programs, ITP enables the integration of self-written programs into the data retrieval, and therefore document production, logically. It furthermore allows developers to fine-tune the data retrieval process for special purposes, such as highly controlled data retrieval, mixed environment/DBMS data retrieval or high-performance data retrieval.

The phase of developing and implementing this abstract view is separated from the creation and execution of models, the document development and usage phase, enabling organisations to limit the data that are accessible from documents. This abstract view can also be modular, so that the DID or parts of the DID can be reused by other collections of documents.

The program that is used for this part of the ITP development process is called the ITP SDK.

## 2.3 Model documents, models and result documents

ITP works with three kinds of files in the text/data merge process:

- **Model documents:** These are word processor documents that can contain all kinds of text with all layout facilities offered by the word processor, mixed with ITP instructions. The ITP instructions enable data retrieval using external retrieval programs or logic, manipulation of data and merging those data into the document text.  
A model document is the equivalent of a program source.
- **ITP Models:** These are binary files generated from model documents, containing an optimised form of the model document, including all text, layout and ITP instructions. Model documents are compiled to ITP models. The model is the equivalent of a program.
- **Result documents:** These are normal word processor documents that contain the final document text, merged with -possibly manipulated- data. Executing ITP models produces these documents.

## 2.4 The ITP document development process

Developing a document in ITP means writing model documents (in your favourite word processor), compiling them to ITP models and testing them.

ITP program logic is written in the model document using the ITP instruction language, a powerful yet easy to use language, which has two very special properties, concerning input/output:

- Input is always done through a highly abstract and structured interface to the database or to external programs. The actual database access is not programmed in the model itself.



Furthermore some user interaction can take place for additional input. All other input for the result document is already present in the model document itself!  
Other means of input are not (and do not have to be) available.

- Output is the result document itself, which means that all output produced is text and layout for the result document(s). Both may of course depend on database content. No special output facilities are needed.

The ITP instruction language furthermore offers all the usual programming facilities: functions, if-then clauses, while-loops, variables, arrays etc. It also offers powerful calculation, data and text manipulation and conversion functions. A model can be designed to produce one or more result documents.

The ITP instruction language is in fact a kind of programming language on word processor document level. Contrary to a normal programming language it is very easy to use (even by trained end-users), because of the special ways in which is dealt with input and output and because of the focus on text and layout in the word processor's native environment.

ITP supports model documents in several word processor formats: Word For Windows 6.0/95/97/2000/XP, WordPerfect 6.1 and up and Ami Pro 4.0. Lotus WordPro is supported through the Ami Pro document format. Support is also available for HTML documents and ASCII documents.

## 2.5 Producing result documents

Executing the corresponding ITP models generates result documents; this is called the RUNMDL process. This process can be initiated from the ITP user interface or it can be integrated in an application, using ITP's APIs.

While running the model users can be asked to make (database) selections or to provide additional input. This is the case with interactive models. Models can also be designed to run without any intervention by the user.

During one run of the model, one or more result documents can be produced.

Performance of ITP document production is very good, determined mostly by how fast data can be retrieved.

The result document is in the same word processor format as the original model document.

## 2.6 Printing ITP output

ITP produces standard word processor documents. These documents are printed using the word processor itself as the print engine, thus allowing the total layout flexibility of modern word processors for your automated output production.

The word processor software uses the operating system's printing architecture to print documents. ITP can therefore be used with all printer-drivers that are available for the operating system. It can - for example- produce Postscript, PCL5, AFP and PDF output, or any other format generated by a Windows printer driver.

This open architecture also enables easy integration of ITP with -for example- fax solutions, web publishing solutions, archiving software and print servers.

## 2.7 The ITP product family

ITP is a product family of document production solutions that consists of the following products:

- ITP/CS for developing and running ITP models for desktop based solutions.
- ITP/Model Development Kit. An ITP/CS extension to completely integrate the document development environment with Microsoft Word 97/2000/XP.



## ITP and XML

- ITP/Batch Server for batch document production. Models for ITP/Batch Server are developed with ITP/CS.
- ITP/Document Services to control document production batch processes, such as running ITP/Batch Server models, printing documents, converting documents, routing documents etc. An introduction white paper can be downloaded from Aia's web site.
- ITP/SDK AS/400 to describe database access for AS/400 databases.
- ITP/SDK MultiPlatform to describe databases access for local access programs, ODBC, Notes/Domino, Oracle, XML data files and mainframe access programs.

Both ITP/CS and ITP/Batch Server support access to multiple database types:

- Lotus Notes Connection
- AS/400 Connection
- Oracle Connection
- ODBC Connection
- XML File Connection
- Mainframe Connection

Data from multiple database sources can be used in a single model.

More information about these products can be found on our web site <http://www.aia-itp.com/>.

White paper



### 3. The ITP Instruction language

The ITP instruction language can roughly be divided into three parts:

1. instructions for the creation of text
2. instructions for data retrieval
3. instructions for data manipulation

To distinguish between text that has to appear in the result document and ITP instructions, the hash symbol # is used as a “tumble switch”; each time a hash symbol is encountered, ITP switches from “instruction-mode” to “text-mode” or vice versa. A model document will always start in text-mode, so before the first instruction a hash symbol has to be placed.

A very simple model document could look like this:

```
#  
BEGIN  
  
#  
This is a very simple model document.  
  
#  
  
END  
#
```

The BEGIN and END instruction indicate the beginning and end of the ITP instruction sequence.

This would result in a document containing the following text:

```
This is a very simple model document.
```

Since no data is being merged, this is of course not a very useful document.

If data has to be merged into the text, this can be done using the @-construct:

```
. . .  
#  
Dear Ms. or Mr. @(Cust.Surname),  
  
In reference to you writing...  
#  
. . .
```



## ITP and XML

In the above example the field *Surname* from the record *Cust* will be merged into the document. But where does this record come from?

Data retrieval is achieved through the use of Entries, abstractions of the database which are defined in the DID. In the following example the entry *Customer* will retrieve data from the database. The fields that are retrieved are grouped in a 'record' that will be referred to as *Cust* and which is available between the ITP instructions DO and OD. The fields that are contained in the record have been defined in the entry definition in the DID. Since the method to actually retrieve the data has also been defined in the DID, no more specifications are needed in the model document.

```
...
WITH Cust IN EXP.Customer DO
#
Dear Ms. or Mr. @(Cust.Surname),

In reference to your writing...
#
OD
...
```

The WITH construct will retrieve one data record. If multiple records should be retrieved you can use the FORALL construct. The DO ... OD part of the FORALL construct will be executed for each record that has been retrieved.

Again, the method to retrieve the data has been defined in the DID.

```
...
WITH Cust IN EXP.Customer DO
#
Dear Ms. or Mr. @(Cust.Surname),

In reference to your writing....

A list of the ordered articles follows:
#
  FORALL Art IN Cust.Ordered_articles DO
#
  @(Art.Number_of_articles) @(Art.Article_description)
#
  OD (* FORALL Art IN Cust.Ordered_articles *)

OD (* WITH Cust IN EXP.Customer *)
...
```

The relation between *Customer* and *Ordered\_articles* has been defined in the DID. So, the developer of model documents only need to know that these relations exist, not how they are implemented.



## ITP and XML

To clarify ITP coding in the model document, comments can be added. Comment starts with (\*) and end with (\*). Most word processors can also aid in the creation of easily readable documents by using different styles, colours, etc.

Of course the contents of a document should differ based on -for example- whether the customer is male or female.

```
...
Dear #
  IF Cust.Sex = "M" THEN # Mr. #
  ELIF Cust.Sex = "F" THEN # Ms. #
  ELSE # Ms. or Mr. #
  FI
# @(Cust.Surname),
```

In reference to your writing...

Although this gets the job done, it is not quite obvious what text will be produced in the result document. Introducing variables will make this somewhat easier.

```
...
TEXT ms_mr

IF Cust.Sex = "M" THEN
  ASSIGN ms_mr := "Mr."
ELIF Cust.Sex = "F" THEN
  ASSIGN ms_mr := "Ms."
ELSE
  ASSIGN ms_mr := "Ms. or Mr."
FI
#
Dear @(ms_mr) @(Cust.Surname),
```

In reference to your writing...

In this example a variable of type TEXT is declared. Then a value is assigned to this variable based on the contents of the field. Finally the contents of the variable are merged into the text using the @-construct. As a result, by separating the logic as much as possible from the text, the structure of the result text can be more *wysiwig*.

More than 60 build-in functions are available to format, convert or otherwise manipulate both numerical and text values.



```
...  
#  
@(Cust.Name)  
@(Cust.Street) @(Cust.Housenumber)  
@(uppercases(Cust.City))  
  
Nijmegen, @(date(today))  
  
...  
#  
...
```

Might result in:

```
Aia Software b.v.  
Kerkenbos 10-129  
NIJMEGEN  
  
Nijmegen, 12 April 1999  
  
...
```

In the above example three built-in functions are used: *uppercases*, *date* and *today*.

*Uppercases* will convert a text into the same text, but -as the name suggests- all into uppercases.

*Today* has no parameters and will result in a numerical representation of the current date. In the above example this is 19990412.

The function *date* will print a date with the name of the month “in words”. Of course the result of the date function depends on the language being used, something that can be changed while producing the text. This is especially useful for internationally operating companies.

It would go beyond the scope of this short introduction to show examples of all the other facilities the ITP instruction language offers.

## 4. XML

### 4.1 A very short introduction to XML<sup>1</sup>

XML is currently one of the most important developments in the IT industry. Everybody is working on integrating XML into his or her product, or at least talking about it. What is XML? XML is nothing more and nothing less than a way to annotate text or data. Applications interpret these annotations. You can use XML, for example, to write a book and define the chapters and sections. This could for example look like this:

```
<CHAPTER>
  <SECTION>
    <PARAGRAPH>
      This is an example of an <EMPHASIS>XML document</EMPHASIS>.
    </PARAGRAPH>
    <PARAGRAPH>
      This is just another paragraph.
    </PARAGRAPH>
  </SECTION>
</CHAPTER>
```

The XML annotations are written between < and >. Such an annotation is called a tag. Every tag must be matched by an end-tag. This tag has the same name as the begin tag but is preceded by an /.

The information enclosed between a begin tag and end tag is called an XML element. An XML element can contain both text and other XML elements, as is shown in the example. The indentations have no meaning in the example, but are added to clarify the structure.

One of the most important things about XML is to realise that the tags in an XML file have no meaning until somebody or something gives it a meaning. The same XML file can mean two completely different things for two applications.

### 4.2 Data exchange with XML

One of the most important reasons for using XML is that XML holds a promise for very versatile data exchanges between different applications and systems. XML is relatively easy to generate and interpret and therefore ideally suited for electronic data interchange.

Tags are used to define *records* and *fields*. Related records are simply nested. Below you will find an example of a **Customer** record with some data fields (both numerical and text fields) and with related **Address** and **Partner** records.

---

<sup>1</sup> The XML specification and related specifications such as XSL and XQL are written by the W3 Consortium. An international standards organisation in which a lot of big players in the industry participate, such as IBM, Microsoft and Adobe. See <http://www.w3.org> for more information.



```
<Customer>
  <CustomerNumber>1002</CustomerNumber>
  <LastName>Benitas-Fuentes</LastName>
  <FirstName>Guanita</FirstName>
  <Initials>G.</Initials>
  <DateOfBirth>19630512</DateOfBirth>
  <DateOfDeath>0</DateOfDeath>
  <Sex>F</Sex>
  <MaritalStatus>M</MaritalStatus>
  <PartnerNumber>1234</PartnerNumber>
  <Address>
    <CustomerNumber>1002</CustomerNumber>
    <ZipCode>1254</ZipCode>
    <Number>123</Number>
    <Street>Gonzales</Street>
    <City>Tijuana</City>
    <Country>Mexico</Country>
  </Address>
  <Partner>
    <CustomerNumber>1234</CustomerNumber>
    <LastName>Benitas</LastName>
    <FirstName>Don</FirstName>
    <Initials>D.A.</Initials>
    <DateOfBirth>19650828</DateOfBirth>
    <DateOfDeath>0</DateOfDeath>
    <Sex>M</Sex>
    <MaritalStatus>M</MaritalStatus>
    <PartnerNumber>1002</PartnerNumber>
  </Partner>
</Customer>
```

### 4.3 ITP and XML

Most connection types for ITP (like ODBC and AS/400) use *data pull* technology, meaning that data is retrieved from a (relational) database management system when needed during the production of documents.

*Data pull* has many benefits, but especially in larger environments and in e-business applications the need for a *data push* mechanism for ITP has arisen. With *data push* the data for ITP is first prepared for the ITP document production process and then the ITP process is started. Some important reasons for using *data push* are:

1. Document production is usually a lot faster when using data push, instead of data pull technology. All data needed is collected at the source and then sent to ITP.
2. The customer is working on a mainframe on which data pull technology is expensive, complex and slow. Data retrieval programs cannot be generated automatically. It would be just as easy for the customer to write a program or query to generate XML files.
3. The customer is using component based design and wants all their components (or applications) to communicate through XML files. ITP is just another component/application.

The ITP XML File Connection is based on data push technology where an application creates an XML file that is offered to ITP for further processing.

## 5. The ITP XML File Connection

### 5.1 ITP Concepts and XML

#### 5.1.1 Fields and entries

ITP has three concepts that play a primary role in the XML File Connection: entries, subentries and fields. The intended use is as follows:

- Any XML element that does not contain other elements is mapped to an ITP field.
- Any XML element that contains other elements is mapped to an ITP entry. The elements contained therein will become either fields or entries, depending on whether they contain other elements or not.

#### 5.1.2 Subentries

```
<insurance-policy>
  <insured>
    <surname>Huinink</surname>
  </insured>
  <billing-address>
    <pobox>112</pobox>
    <zipcode>9999 XX</zipcode>
    <city>Nijmegen</city>
    <country>Netherlands</country>
  </billing-address>
</insurance-policy>
```

All relations between entities in the XML file are expressed in the structure of the XML file. The entries that are contained within other entries will become subentries.

#### 5.1.3 Plural and singular entries and optional elements

The XML specifications state that every XML file has exactly one root element. The main entry of an XML DID is therefore always a singular entry. In XML definitions, so-called DTD's or Schema's, it is possible to define that certain elements can occur at most once, exactly once, more than once or more than zero times. In ITP this is easily expressed as singular (WITH) and plural (FORALL) entries.

#### 5.1.4 The XML input file

When running an ITP model that uses the XML File Connection the XML input file needs to be known. Therefore this input file can be passed as a parameter to the ITP process. Document production then retrieves the needed data from this file.

It is actually also possible to mix access to XML data files with access to other types of data, but that goes beyond the scope of this WhitePaper.



## 5.2 The DID

### 5.2.1 Entries

In the DID you specify what tag name corresponds with a specific Entry (keyword DATA\_RETRIEVAL in the DID Document). These tags contain nested tags: sub-entries or fields. The root element of the XML file is always the (only) main entry for an XML connection DID.

### 5.2.2 Fields

An entry specifies a list of fields to retrieve from the XML data. The DID developer decides which fields should be retrieved. The fields are defined with their tag name (keyword DATABASE\_FIELD). Fields contain no nested tags.

## 5.3 Examples

### XML File

```
<Customer>
  ...
  <CustomerNumber>...</CustomerNumber>
  <LastName>...</LastName>
  <FirstName>...</FirstName>
  ...
  <Partner>
    ...
    <CustomerNumber>...</CustomerNumber>
    <LastName>...</LastName>
    <FirstName>...</FirstName>
    ...
  </Partner>
  <Address>
    ...
    <Street>...</Street>
    <City>...</City>
    ...
  </Address>
</Customer>
```



## DID

```
DEFINE_ENTRY
  NAME Customer
  MODEL_DOCUMENT_STATEMENT WITH
  DATA_RETRIEVAL "Customer"
  DEFINE_FIELDS
    ...
    Customernumber DOUBLE DATABASE_FIELD "CustomerNumber"
    Lastname C_CHAR LENGTH(255) DATABASE_FIELD "LastName"
    Firstname C_CHAR LENGTH(255) DATABASE_FIELD "FirstName"
    ...
  END_DEFINE_FIELDS
  DEFINE_SUBENTRIES
    Address
    Partner
  END_DEFINE_SUBENTRIES
END_DEFINE_ENTRY
```

## 5.4 The XML File Connection Wizard

Defining a DID as shown in the example above would be quite cumbersome. The ITP/SDK Multi Platform therefore contains a wizard to generate a DID from a sample XML file. As long as the structure of the XML file is the same as the structure of the files that will be used in production, the generated DID will be complete. Tags are mapped to ITP names for either entries or fields. Using the wizard means hardly any work at all to define the DIDs.

## 5.5 Conformance to XML standards

XML files offered to ITP should be well-formed, including denotations for character encoding. The ITP XML File Connection strictly conforms to the XML standards with few exceptions. The most notable exception is that ITP assumes that numbers are written with a . as the decimal separator, but an override for this default behaviour can be configured.

### 5.5.1 Character encodings

XML files can use various character 'encodings'. The encoding specifies how to map the physical characters in the file to Unicode. XML files for ITP need to be encoded as UTF-8 (for Unicode), ASCII or ISO-8859-1 (Latin 1).



## 6. Conclusion

XML data and ITP are a very good match for batch document productions (ITP/Batch Server and ITP/Document Services). Performance is very good, data definition is very versatile and the actual back office systems delivering the data are only loosely coupled to the document production system. This may also be considered an advantage in individual document productions (ITP/CS). The disadvantage of the XML File Connection compared to other ITP Connections is that somehow the XML data files have to be generated and have to be sent to the server (ITP/Batch Server) or client (ITP/CS) where ITP is running.



ITP is developed by **Aia**  
Software

For more information please contact us.

Telephone: +31 24 371 02 30  
Fax: +31 24 371 02 31  
WWW: <http://www.aia-itp.com>  
Email: [itp@aia-itp.com](mailto:itp@aia-itp.com)  
Postal Address: P.O. Box 38025  
6503 AA Nijmegen  
The Netherlands  
Visiting Address: Kerkenbos 10-129  
6546 BJ Nijmegen  
The Netherlands