

# An XML Based Report Writer Component

Iping Supriana, Reza Ferrydiansyah, Wikan Danar Sunindyo

Informatics Department, Institut Teknologi Bandung, Indonesia

[iping@informatika.org](mailto:iping@informatika.org), [reza@informatika.org](mailto:reza@informatika.org), [wikan@informatika.org](mailto:wikan@informatika.org)

## Abstract

The report is commonly used as a tool to represent and organize data on a medium, commonly paper, such that only relevant information is shown and obtained by an entity having a need for it. The current line of computer based report writers gathers data from a database and prints the relevant information to the user. The layout and the filtering of data is still the responsibility of the user.

We have implemented a report writer component, which uses XML (eXtended Markup Language) to represent the report's layout and data filter. In this paper we will describe the report writer component, the reason XML is chosen as the representation, and show how it is possible to represent a report through XML

**Keywords :** Report Writer, System Information, XML

## 1. Introduction

The report is commonly used as a tool to represent and also organize data on a medium, commonly paper, such that only relevant information is shown and obtained by an entity having a need for it. A well-designed report is an essential part of a computer based information system, as a report can effectively summarize important information from multiple data sources.

We define the report writer as an application which takes data from a database, builds the report, and outputs it to the user (e.g. via the printer or monitor).

The building process of reports in a report writer software usually consists of two steps. One is building the report's database query, which will determine the data shown in the report. The next step will be building the report's layout; this will determine where the position of each data and other relevant objects is to be shown in the final report.

Since the same report can be used over and over (albeit with different data), the report layout and queries must be stored in an internal representation which can be easily written to and interpreted by the software. We are currently developing Repro, a report writer which uses XML (version 1.0) as the internal representation of the report.

The architecture of Repro is seen in figure 1. Repro consists of two main parts, the layout/query builder which creates the XML file and the interpreter which generates the report based on the layout and on the data contained in the database.

In this paper we will describe how XML is used to represent the layout and the query of the report which can be generated by Repro.

## 2. Related Work

Repro was inspired by Crystal Report[1], a popular report builder software which could be integrated into many solutions. To use crystal report, a software engineer creates the layout of the report in

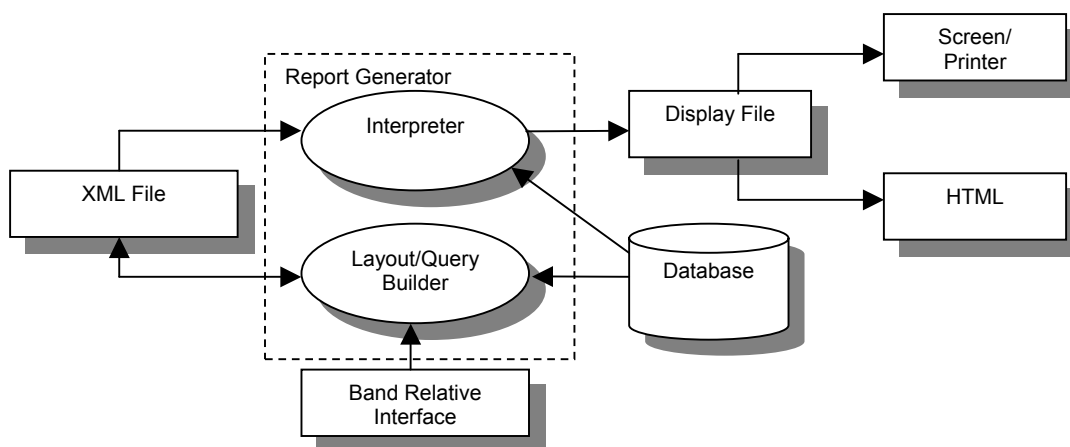


Figure 1 : Architecture of Repro

the user interface of the Crystal Report. He will then save the report format into a file, and finally link an application with the Crystal Report library (dll).

Report builder software such as crystal report has

'proprietary' representation of its report. A user must own a copy of Crystal Report if he wants to change or create a new report included in a database.

### 3. Application Description

As part of the RUSNAS, research funded by the Indonesia's ministry of technology, we at the department of Informatics, Institut Teknologi Bandung has created the report writer Repro.

As shown on figure 1, there are two distinct parts of Repro, the layout and query builder and the interpreter.

The layout and query builder of Repro is a user interface which is used to build the layout of the report. The layout and query builder will create an XML file which represents the layout and the query of the report being prepared.

The second part is the interpreter, the interpreter takes an XML file and parses it. It will then create the layout and uses the query to retrieve data from the database and creates a report like that shown in figure 2.

No.	LEV	PIL	SISTEM		TAMPIL	PROGRAM		DESKRIPSI
			HAMA	AHAK		ADRES	NPG	
1	3	3	Abdi	Layan Lembaga	MF2b	MASTER	0	Memberi pelayanan kepad masyarakat, berdasarkan Kemampuan Lembaga
2	3	2		Layanan Ahli	MF2a	MASTER	0	Memberi pelayanan kepad masyarakat, berdasarkan Bidang Keahlian
3	3	1		Penyuluhan	MF1	MASTER	0	Memberi Latihan, Penyuluhan pada Masyarakat
4	3	4		Buku SMA	MF3	MASTER	0	Mensusunuln karya pengabdian pada masyarakat termasuk buku
10								
5	3	1	Dikjar	lazah	AF1	MASTER	0	Memperoleh lazah
6	3	12		Panitia UJ	AF7	MASTER	0	Bertugas dalam Panitia Ujian Akhir
7	3	16		Diklat	AF11	MASTER	0	Membuat Diklat
8	3	15		Dosen muda	AF10	MASTER	0	Membina Tenaga Pengajar yang lebih muda
9	3	14		Calon Dosen	AF9	MASTER	0	Membimbing Tenaga Pengajar dalam rangka Studi S2
10	3	13		Bina Mhs	AF8	MASTER	0	Membina Kegiatan Mahasiswa
11	3	2		Kuliah	AF2a	MASTER	0	Memberi Kuliah pada Fakultas Sendiri
12	3	3		Fakultas	AF2b	MASTER	0	Memberi Kuliah pada Fakultas di PT sendiri tiap semester

Figure 2 : Example of Report

### 4. Repro's XML Format

The XML (eXtensible Markup Language) format[2] is an emerging format which has gained popularity in recent times. XML is a markup language such as HTML and uses tags in the form of <tagname>..</tagname> as its primary representation of data. There are may examples of a conversion to XML from older formats. [find the thing]

There are several reason why we use XML to represent the report for Repro :

1. XML is extensible, we were free to add any tags to our format, and thus we did not have to constraint ourselves with available tags.

2. XML is standardized and universally recognized. Therefore our tag system would be easy to learn, and easy to implement. A user could choose not to use Repro's layout and query builder software and use a different software to create the report's layout.
3. By using XML as the basis of our representation, a user can change the layout of the report via the XML text instead of having to use if there is a need to change the report format for some reason.

For our representation we divide a report into 5 sections:

1. Page Header. This section is a text or image shown in the top of all pages. A page header can be divided into rows and columns. This part can also consists of special data such as the page number and the filename
2. Header. This section consists of a text or image, shown once at the beginning of the report. This section can also be divided into rows and columns.
3. Body. The body section is the section where data from the database is shown, arranged into a columnar table.
4. Tail. This section is a text or image which is shown once at the end of the report which can be divided into rows and columns.
5. Page Tail, this section is like the page header, only that it is shown at the bottom of the page.

Each section is represented as a set of <table> tags which has a respective ID of PH, Head, Body, Tail, and PT. For the head, page head, tail and page tail section the <table> part is similar to that of the HTML [3] counterpart.

Each cell element (the lowest <td> tag) can hold one type of text (one font and one size only) to simplify the interface. The controllable attributes for each cell elements are size, background color, font type, and font size.

The Body section is also represented as a set of <table> tags which has an ID of Body. In this section each <tr> tag is printed as one table column. The content in the first <td> tag is the location of the database relative to a user's computer. The third <td> tag shows the query used to retrieve the data.

The second <td> tag represents the data layout of the report. Each field is given its own tag <fi> with its corresponding attributes (width, special calculation, id).

Each field may also have their own headers which can also be joined with other column headers to give super headers. An Example of this is shown in figure 2, the column headers 'NAMA' and 'ANAK' is joined to create a super header 'ITEM'. This is also controlled in its attribute of fi.

```

<!ELEMENT report (table)+>
<!ATTLIST report
  col2 CDATA #REQUIRED
  next CDATA #REQUIRED
  repeat (yes/no) #REQUIRED>

<!ELEMENT table (tr)+>
<!ATTLIST table i (PH|HEAD|BODY|TAIL|PT)
#REQUIRED>

<!ELEMENT tr (td)+>
<!ATTLIST tr i CDATA #REQUIRED>

<!ELEMENT td (CDATA|(ix,te,fi)+>
<!ATTLIST td
  i CDATA #REQUIRED
  wid CDATA #REQUIRED
  fn CDATA #IMPLIED
  sz CDATA #REQUIRED
  st CDATA #IMPLIED
  co CDATA #REQUIRED
  bg CDATA #REQUIRED
  box CDATA #IMPLIED
  border CDATA #IMPLIED>

<!ELEMENT ix EMPTY>
<!ATTLIST ix
  ix index CDATA #REQUIRED>

<!ELEMENT te EMPTY>
<!ATTLIST te
  a CDATA #IMPLIED
  b CDATA #IMPLIED
  c CDATA #IMPLIED
  d CDATA #IMPLIED>

<!ELEMENT fi (CDATA)+>
<!ATTLIST fi
  ts CDATA #REQUIRED
  field CDATA #REQUIRED
  width CDATA #REQUIRED
  opsi CDATA #IMPLIED>

```

**Table 1.** DTD for XML Representation

The complete DTD of Repro's XML Representation is shown in table 1.

## 5. Conclusion

In this paper we have shown how XML can be used to represent a report built via a report writer

application. One of the reason XML was chosen as the internal representation of the report is because it is flexible and universally recognized. By using XML, we hope to achieve operability with other software meaning that the report can be built with software other than Repro.

## 6. Reference

1. Seagate, **Crystal Report** [On-line], <http://www.crystaldecisions.com/products/crystalreports/>, 1 September 2001
2. W3C, **XML Specification** [On-line], <http://www.w3.org/XML>, 15 September 2001.
3. W3C, **HTML 4.0 Specification** [On-line] <http://www.w3.org/HTML>, 3 July 2002